Popma

Minh Chau Do

09 10 2023

Daten von Popma

```
#Datensortierung

# In Mortalität umrechnen
popma_r$V2 <- 100 - popma_r$V2

popma_r <- popma_r[complete.cases(popma_r),]

popma_r <- popma_r[order(popma_r$V2), ]

# Werte außerhalb des Bereichs [0, inf] raus
popma_r <- popma_r[popma_r$V1 >= 0, ]
xr_popma <- sort(popma_r$V1)
yr_popma <- popma_r$V2



# In Mortalität umrechnen
popma_b$V2 <- 100 - popma_b$V2

popma_b <- popma_b[complete.cases(popma_b),]

popma_b <- popma_b[order(popma_b$V2), ]

# Werte außerhalb des Bereichs [0, inf] raus
popma_b <- popma_b[popma_b$V1 >= 0, ]
xb_popma <- sort(popma_b$V1)
yb_popma <- popma_b$V2
```

Startfunktion

Um die bestmögliche Anpassung an den vorliegenden Datenpunkten darzustellen wurde mithilfe einer Funktion der beste Startparameter mit dem kleinsten Fehler errmittelt, da die Anpassung stark von den initialen Startwerten abhängig ist.

```
find_best_start_2parameter <- function(p, q, max_beta, max_eta, steps_beta,
                                       steps_eta, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 2)  # Matrix für Startparameter
```

```
  for (beta in seq(0, max_beta, by = steps_beta)) {
    for (eta in seq(0, max_eta, by = steps_eta)) {
      start_params <- c(beta, eta)

      # Schätze die Parameter mit den aktuellen Startparametern
      if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
        output <- capture.output({
        result <- getstuexp2(
                        p = p, q = q, start = start_params,
                        show.output = TRUE, plot = FALSE, wert1 = 2
                      )
        })
      }
      else{
        output <- capture.output({
        result <- fitting_function(p = p, q = q, start = start_params,
                                   show.output = TRUE, plot = FALSE)
        })
      }

      # Berechne den Fehler (popma_r_1$value) für die aktuellen Startparameter
      current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

      # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
      if (!is.na(current_error)) {
        best_errors <- c(best_errors, current_error)
        best_starts <- rbind(best_starts, start_params)
      }
    }
  }


  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}


find_best_start_3parameter <- function(p, q, max_shape1 = 10, max_shape2 = 10,
                                       max_scale = 10, steps_shape1, steps_shape2,
                                       steps_scale, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 3)  # Matrix für Startparameter
```

```r
  for (shape1 in seq(0, max_shape1, by = steps_shape1)) {
    for (shape2 in seq(0, max_shape2, by = steps_shape2)) {
      for (scale in seq(0, max_shape1, by = steps_scale)) {
        start_params <- c(shape1, shape2, scale)

        # Schätze die Parameter mit den aktuellen Startparametern
        if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                          p = p, q = q, start = start_params,
                          show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
      }

        else{
          output <- capture.output({
            result <- fitting_function(p = p, q = q, start = start_params,
                                show.output = TRUE, plot = FALSE)
          })
        }
        # Berechne den Fehler (popma_r_1$value) für die aktuellen Startparameter
        current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

        # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
        if (!is.na(current_error)) {
          best_errors <- c(best_errors, current_error)
          best_starts <- rbind(best_starts, start_params)
        }
      }
    }
  }

  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}




find_best_start_4parameter <- function(p, q, max_shape, max_scale, max_rate,
                                       max_mix, steps_shape, steps_scale,
                                       steps_rate, steps_mix,fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
```

```
  best_starts <- matrix(nrow = 0, ncol = 4)  # Matrix für Startparameter

  for (shape in seq(0, max_shape, by = steps_shape)) {
    for (scale in seq(0, max_scale, by = steps_scale)) {
      for (rate in seq(0, max_rate, by = steps_rate)) {
        for (mix in seq(0, max_mix, by = steps_mix)) {
          start_params <- c(shape, scale, rate, mix)

          # Schätze die Weibull-Parameter mit den aktuellen Startparametern
          output <- capture.output({
            result <- fitting_function(p = p, q = q, start = start_params,
                                       show.output = TRUE, plot = FALSE)
          })

          # Berechne den Fehler (popma_r_1$value) für die aktuellen Startparameter
          current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

          # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
          if (!is.na(current_error)) {
            best_errors <- c(best_errors, current_error)
            best_starts <- rbind(best_starts, start_params)
          }
        }
      }
    }
  }

  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}
```

Datenanpassung an die Daten von Popma

Weibullverteilung

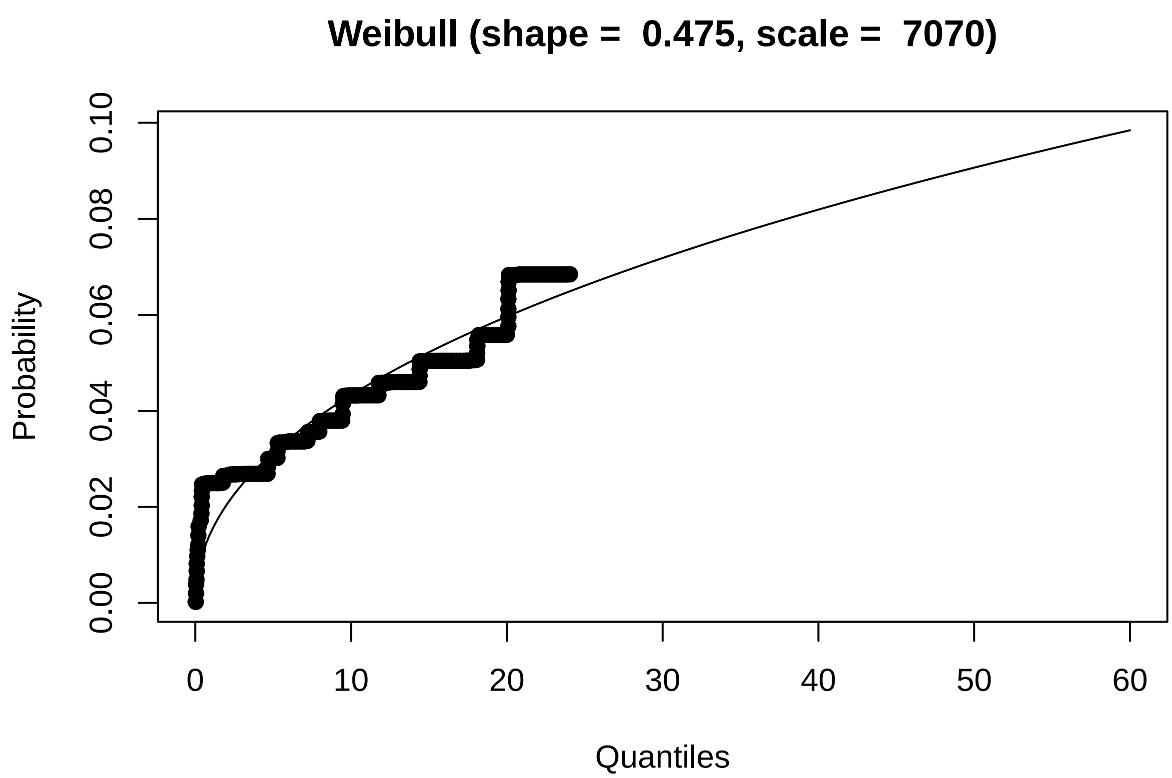
```
# Weibullverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibullpar.R")

best_popma_r_1 <- find_best_start_2parameter(p = yr_popma/100, q = xr_popma,
                                             max_beta = 10, max_eta = 10,
                                             steps_beta = 1, steps_eta = 1,
                                             fitting_function = getweibullpar)
```

```
## Bester Startparameter: 2 1
```

```
## Bester Fehler: 1.316567e-07
```

```r
popma_r_1 <- getweibullpar(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_popma_r_1,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1]    0.4752779 7074.9589227
##
## $value
## [1] 1.316567e-07
##
## $counts
## function gradient
##       75       75
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

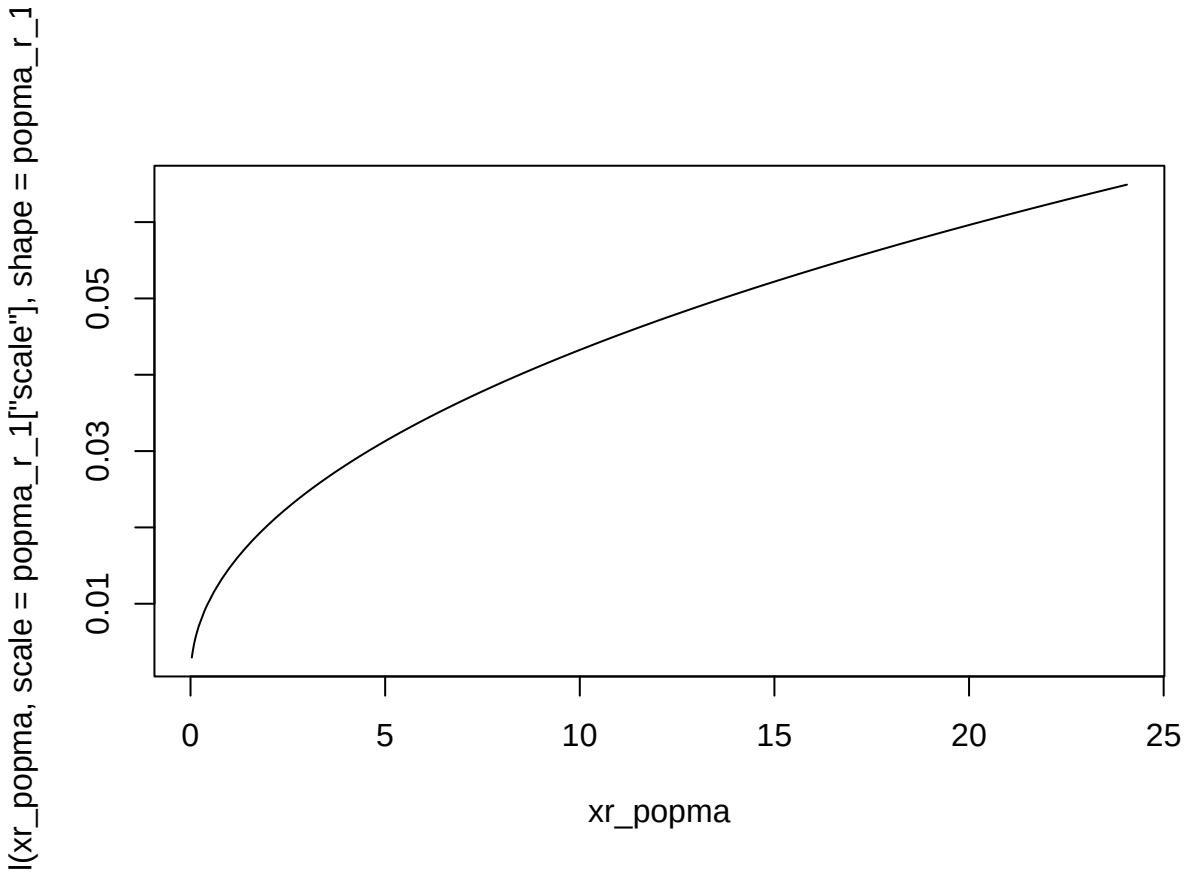


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
popma_r_1
```

```
##        shape        scale
##    0.4752779 7074.9589227
```

```
plot(xr_popma,
     pweibull(xr_popma,
              scale = popma_r_1["scale"],
              shape = popma_r_1["shape"]), type = "l")
```



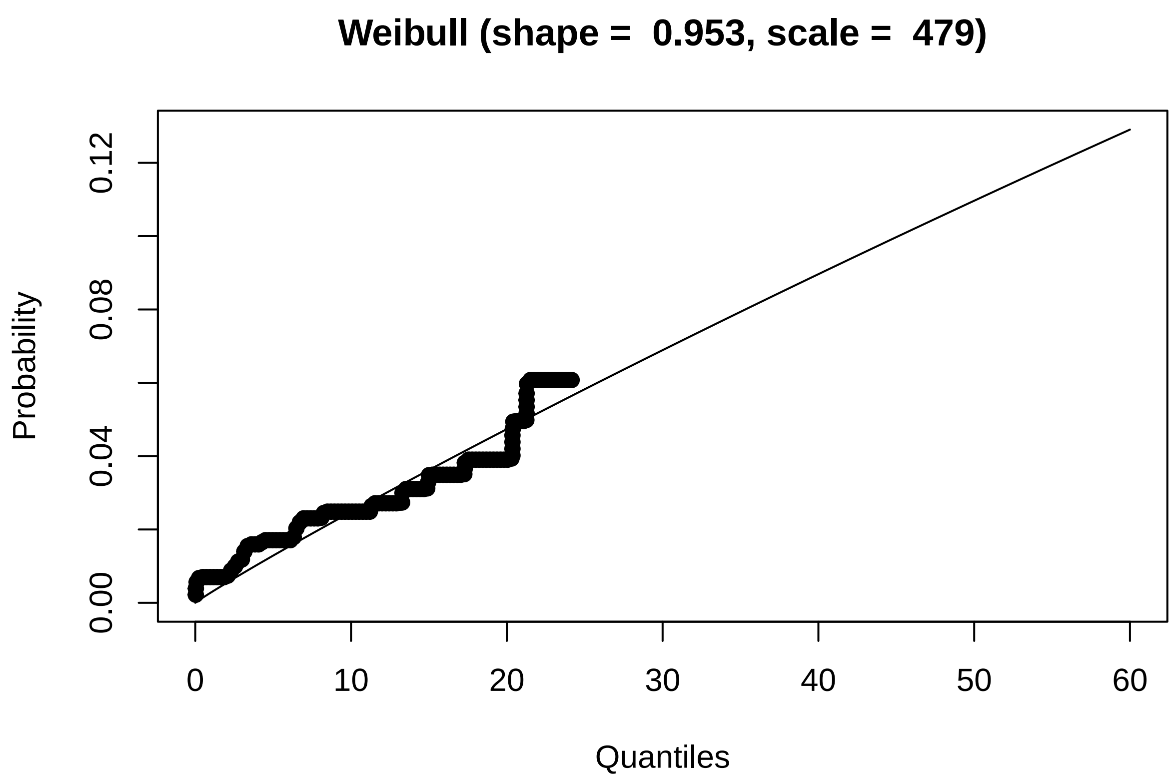

```
best_popma_b_1 <- find_best_start_2parameter(p = yb_popma/100, q = xb_popma,
                                             max_beta = 10, max_eta = 10,
                                             steps_beta = 1, steps_eta = 1,
                                             fitting_function = getweibullpar)
```

```
## Bester Startparameter: 7 3
## Bester Fehler: 1.71587e-07
```

```
popma_b_1 <- getweibullpar(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_popma_b_1,
                        show.output = TRUE,
                        plot = TRUE
                       )
```

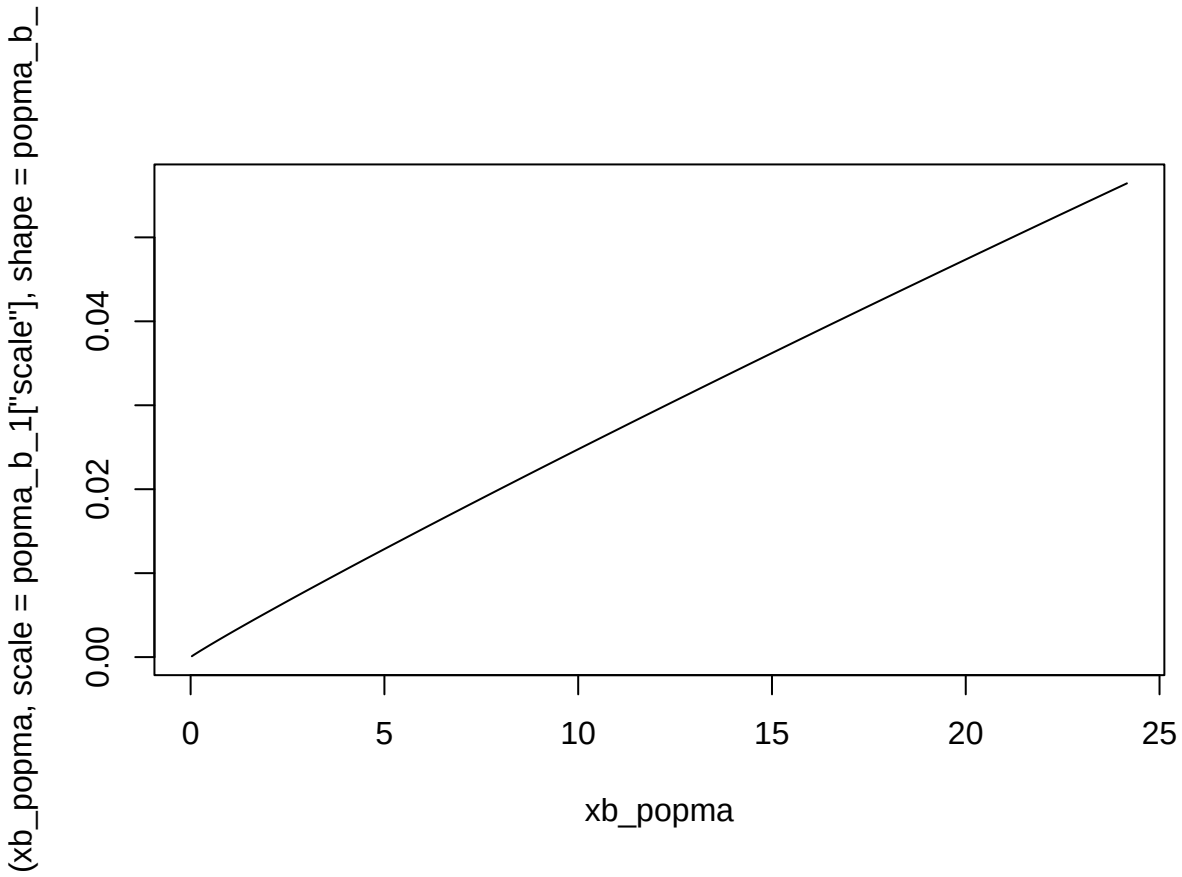
```
## $par
## [1]   0.9526243 479.1299696
```

```
##
## $value
## [1] 1.71587e-07
##
## $counts
## function gradient
##       57       57
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
plot(xb_popma,
     pweibull(xb_popma,
              scale = popma_b_1["scale"],
              shape = popma_b_1["shape"]), type = "l")
```



Exponentiierte Weibullverteilung

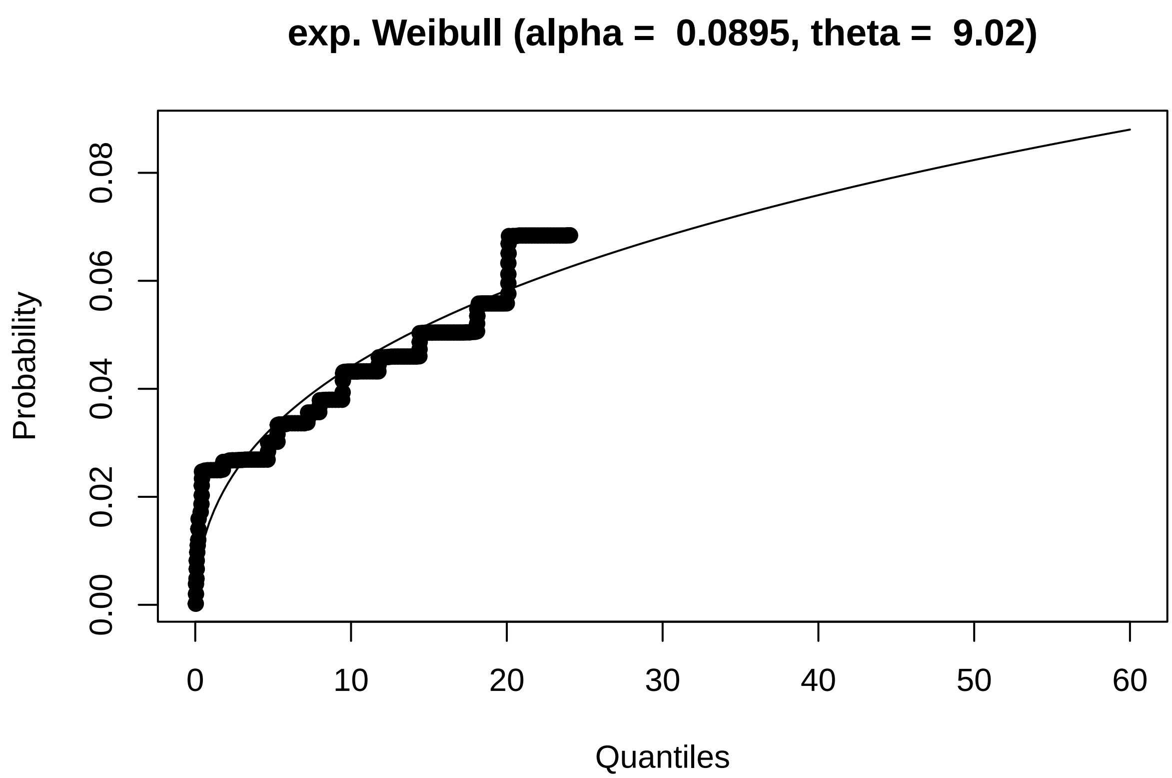
```
# exponentiierte Weibullverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibpar.R")

best_weibbull_xr_popma <- find_best_start_2parameter(p = yr_popma/100,
                                                     q = xr_popma,
                                                     max_beta = 10,
                                                     max_eta = 10,
                                                     steps_beta = 1,
                                                     steps_eta = 1,
                                                     fitting_function = getweibpar)
```

```
## Bester Startparameter: 0 3
## Bester Fehler: 1.44621e-07
```

```
weibbull_xr_popma <- getweibpar(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_weibbull_xr_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.08953179 9.01668227
```

```
##
## $value
## [1] 1.44621e-07
##
## $counts
## function gradient
##       16       16
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

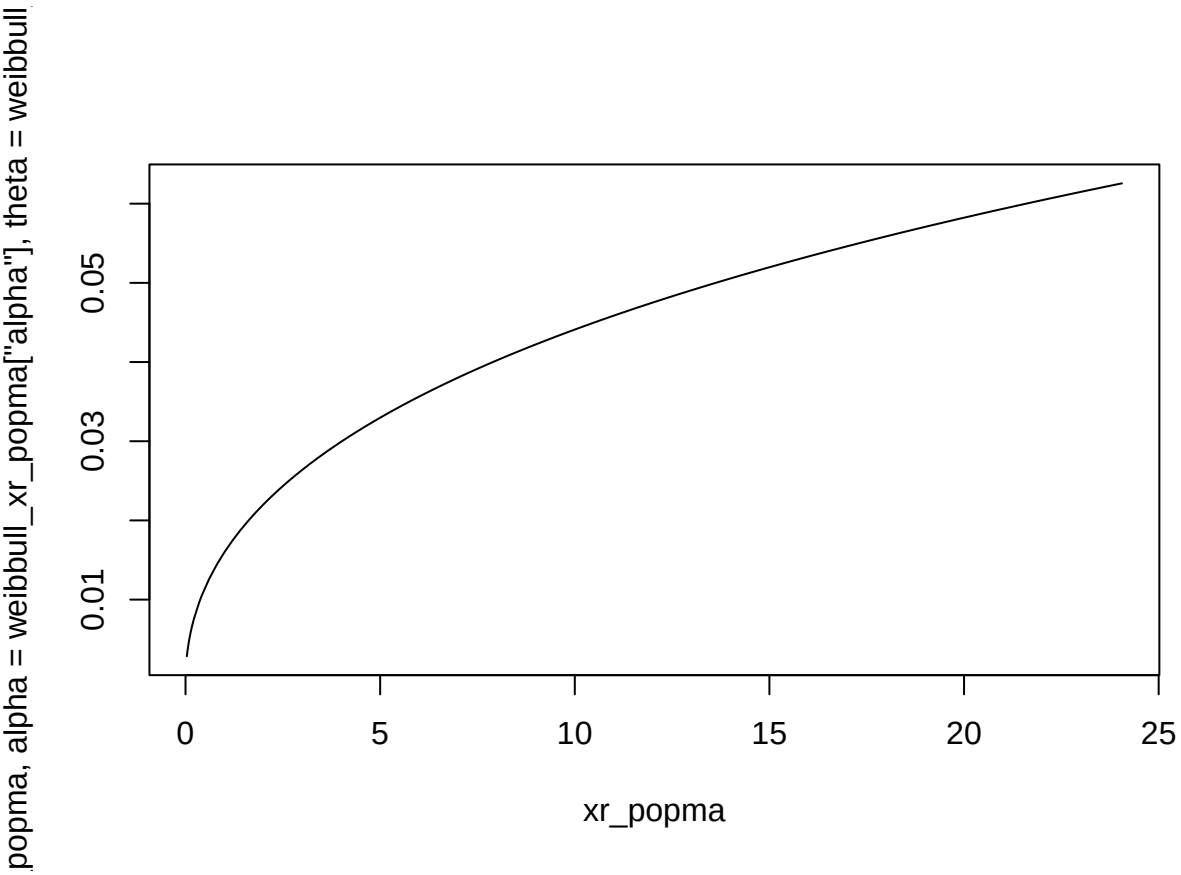


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
weibbull_xr_popma
```

```
##      alpha      theta
## 0.08953179 9.01668227
```

```
plot(xr_popma,
     pexpo.weibull(xr_popma,
              alpha = weibbull_xr_popma ["alpha"],
              theta = weibbull_xr_popma ["theta"]), type = "l")
```



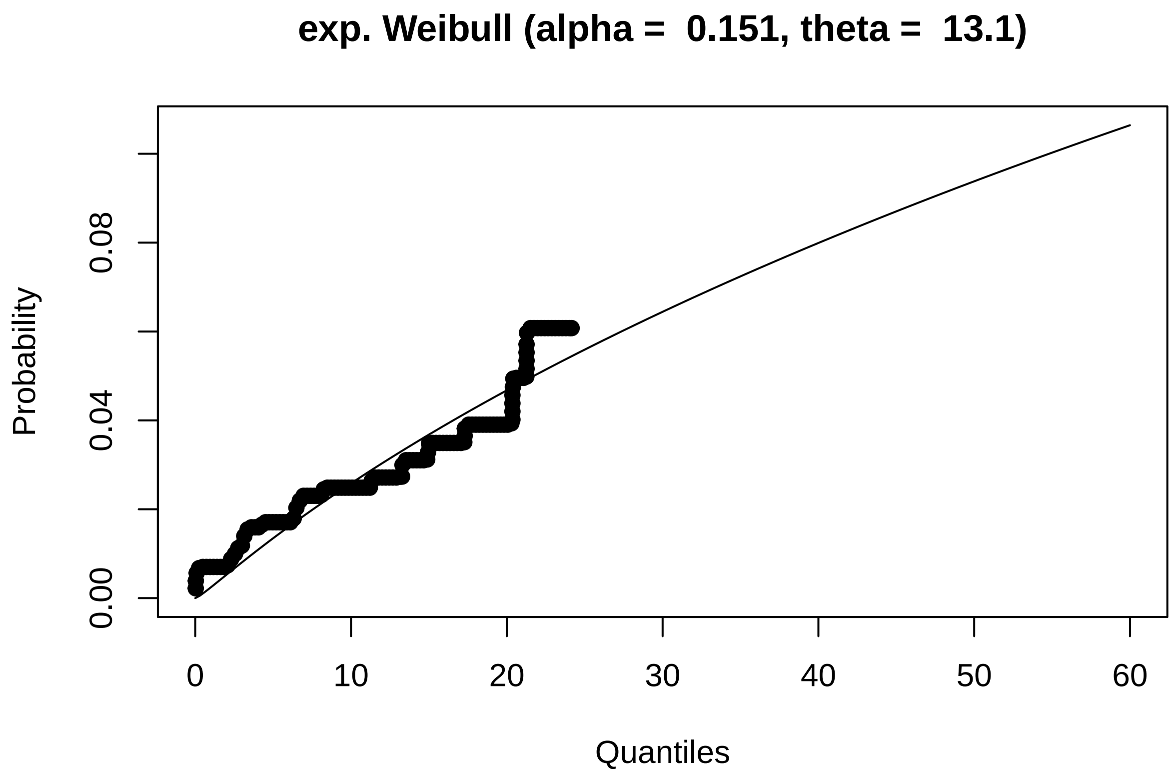

```
best_weibbull_xb_popma <- find_best_start_2parameter(p = yb_popma/100,
                                                     q = xb_popma,
                                                     max_beta = 10,
                                                     max_eta = 10,
                                                     steps_beta = 1,
                                                     steps_eta = 1,
                                                     fitting_function = getweibpar)
```

```
## Bester Startparameter: 0 4
## Bester Fehler: 1.923822e-07
```

```
weibbull_xb_popma  <- getweibpar(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_weibbull_xb_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1]  0.1505674 13.1302101
##
## $value
## [1] 1.923822e-07
##
## $counts
## function gradient
```

```
##       21       21
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

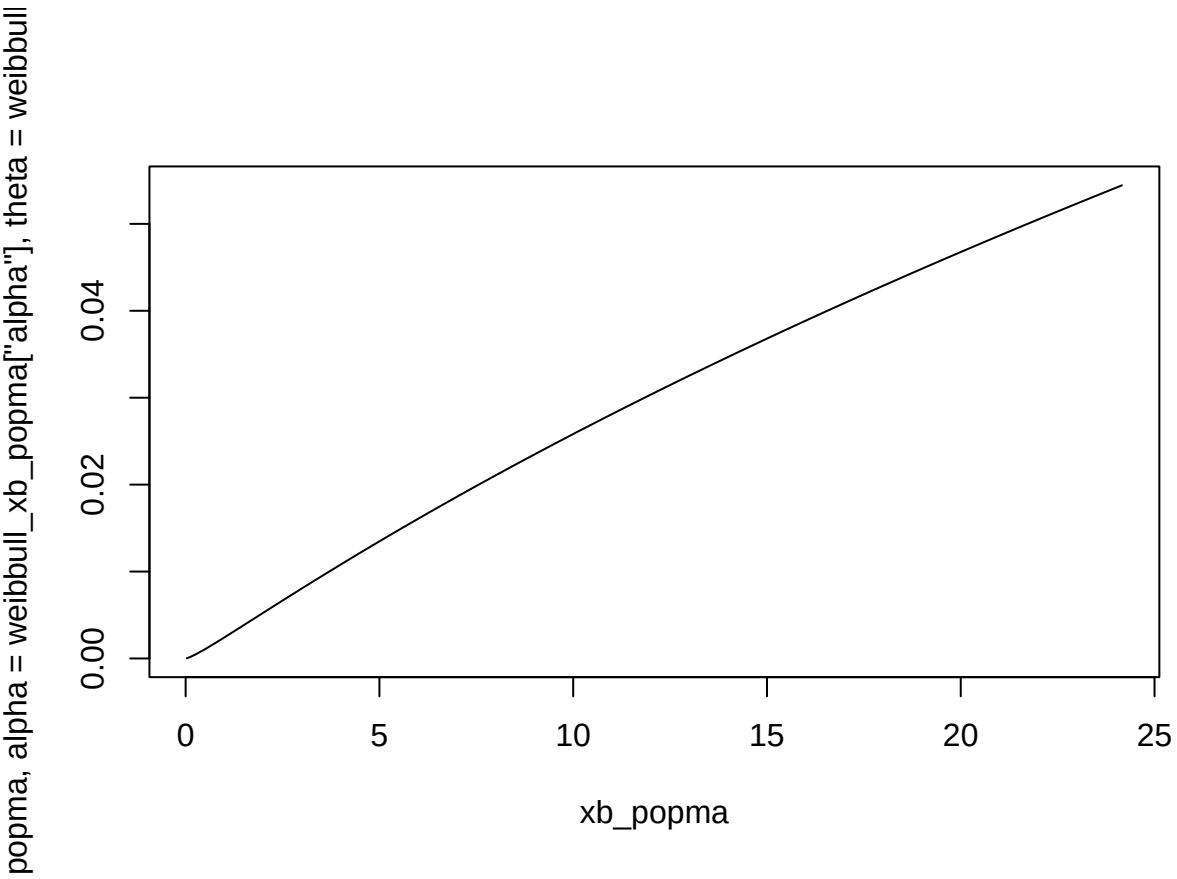


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```r
weibbull_xb_popma
```

```
##      alpha      theta
##  0.1505674 13.1302101
```

```r
plot(xb_popma,
     pexpo.weibull(xb_popma,
              alpha = weibbull_xb_popma ["alpha"],
              theta = weibbull_xb_popma ["theta"]), type = "l")
```



Mischung aus Weibull- und Exponentialverteilung

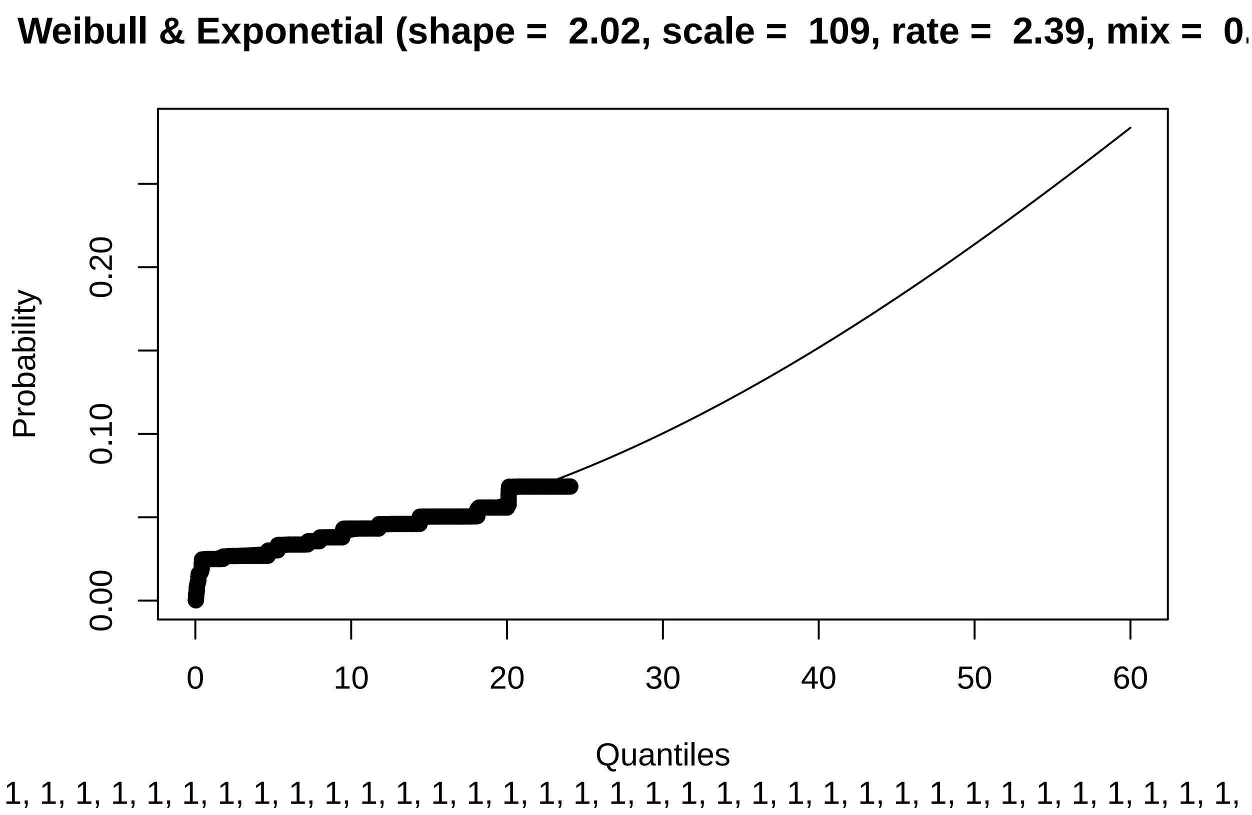
```
# Mischung aus Weibull- und Exponentialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getweibex.R")

best_weibex_xr_popma <- find_best_start_4parameter(p = yr_popma/100,
                                                   q = xr_popma,
                                                   max_shape = 1,
                                                   max_scale = 100,
                                                   max_rate = 0.7,
                                                   max_mix = 1,
                                                   steps_shape = 0.1,
                                                   steps_scale = 20,
                                                   steps_rate = 0.1,
                                                   steps_mix = 0.1,
                                                   fitting_function = getweibex)
```

```
## Bester Startparameter: 0.5 20 0.7 0.4
## Bester Fehler: 6.242942e-08
```

```
weibex_xr_popma <- getweibex(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_weibex_xr_popma, # c(0.5, 60, 0.4, 0.1),
                        show.output = TRUE,
                        plot = TRUE
```

```
                       )
```

```
## $par
## [1]   2.016402 108.529717   2.392628   3.460866
##
## $value
## [1] 6.242942e-08
##
## $counts
## function gradient
##       54       54
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

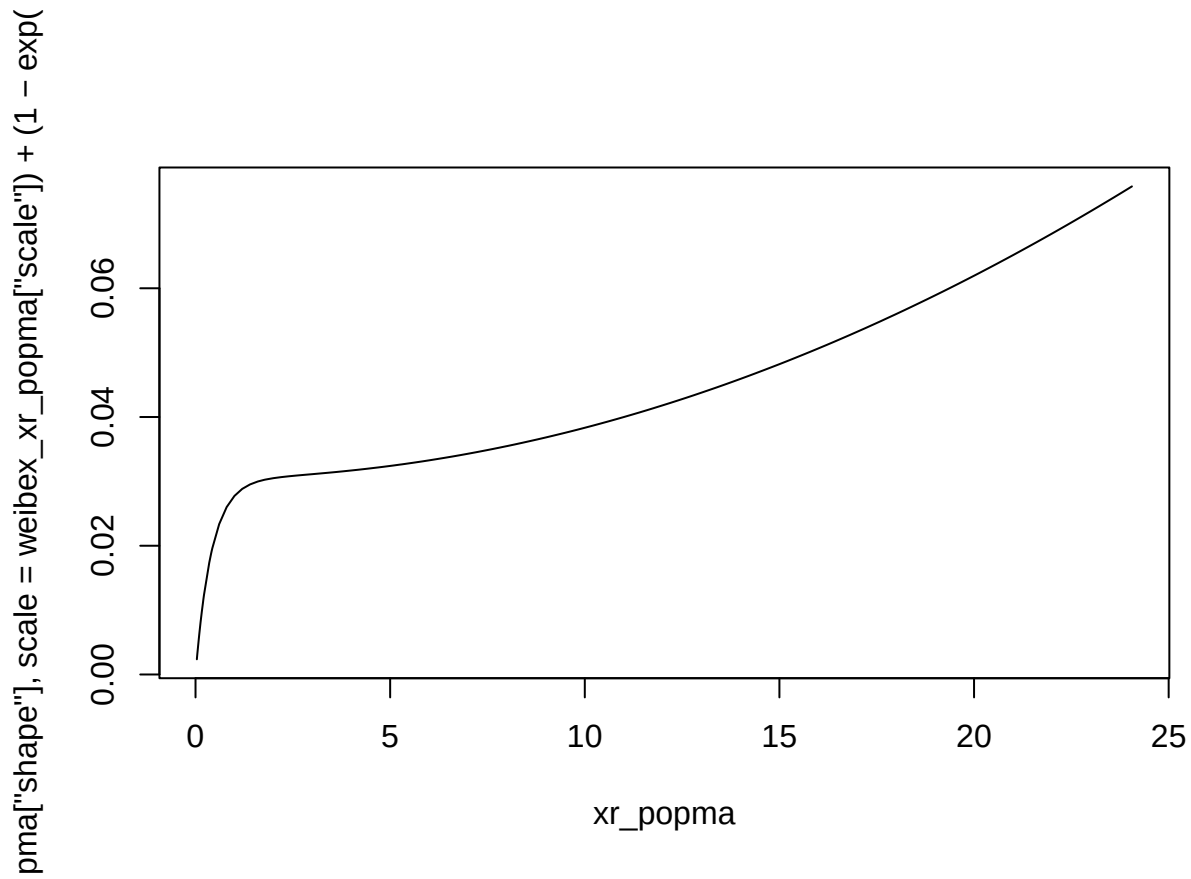



```
weibex_xr_popma
```

```
##      shape      scale       rate        mix
##   2.016402 108.529717   2.392628   3.460866
```

```
# Wenn Ergebnisse aus weibex_xr1_1 von Funktion abgelesen
plot(xr_popma,
     (exp(weibex_xr_popma["mix"]) / ( 1 + exp(weibex_xr_popma["mix"])) *
        stats::pweibull(q = xr_popma,
```

```
                        shape = weibex_xr_popma["shape"],
                        scale = weibex_xr_popma["scale"]) +
       (1 - exp(weibex_xr_popma["mix"]) / ( 1 + exp(weibex_xr_popma["mix"]))) *
       stats::pexp(q = xr_popma,
                   rate = weibex_xr_popma["rate"])),
     type = "l")
```



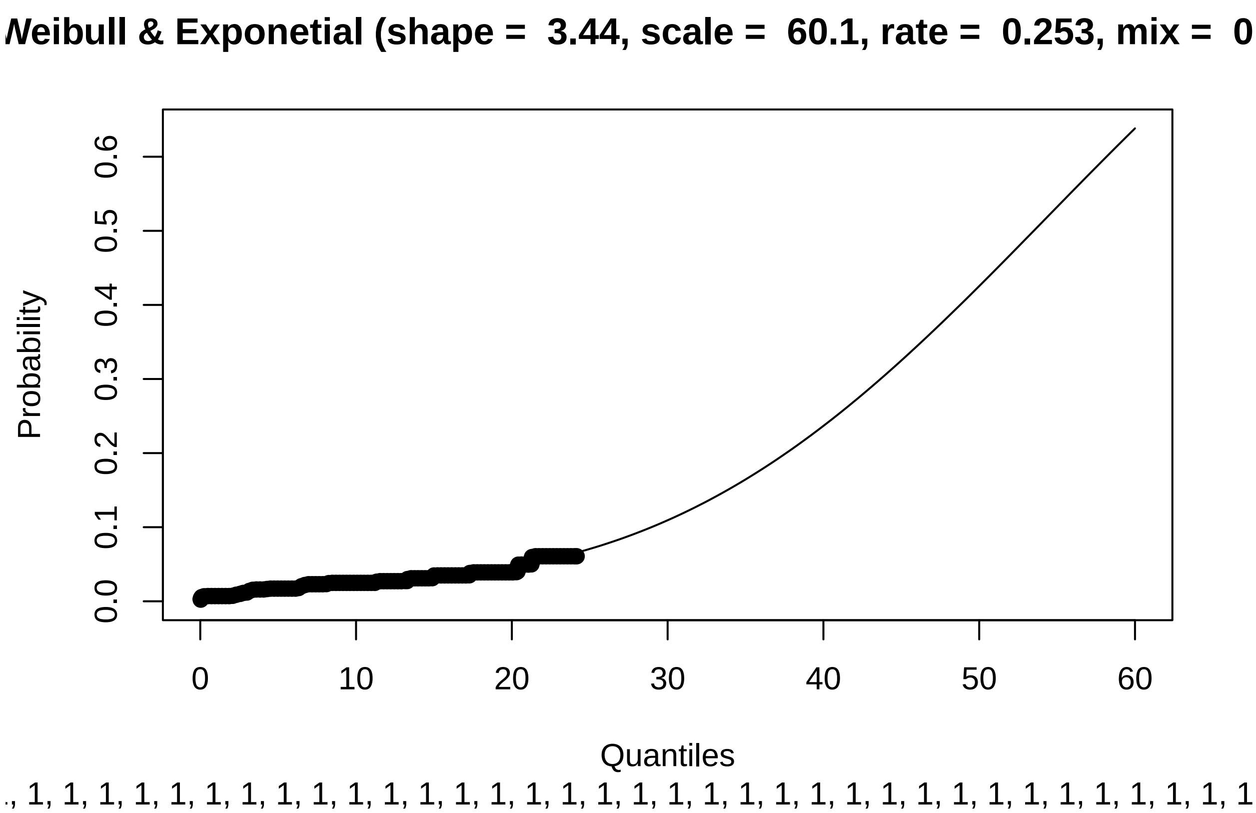

```
best_weibex_xb_popma <- find_best_start_4parameter(p = yb_popma/100,
                                                   q = xb_popma,
                                                   max_shape = 1,
                                                   max_scale = 100,
                                                   max_rate = 0.7,
                                                   max_mix = 1,
                                                   steps_shape = 0.1,
                                                   steps_scale = 20,
                                                   steps_rate = 0.1,
                                                   steps_mix = 0.1,
                                                   fitting_function = getweibex)
```

```
## Bester Startparameter: 0.3 60 0.7 0.3
## Bester Fehler: 6.267981e-08
```

```
weibex_xb_popma <- getweibex(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_weibex_xb_popma, # c(0.5, 60, 0.7, 0.1),
                        show.output = TRUE,
```

```
                        plot = TRUE
                      )
```

```
## $par
## [1]  3.4449757 60.1381595  0.2532234  3.6880388
##
## $value
## [1] 6.267981e-08
##
## $counts
## function gradient
##       19       19
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

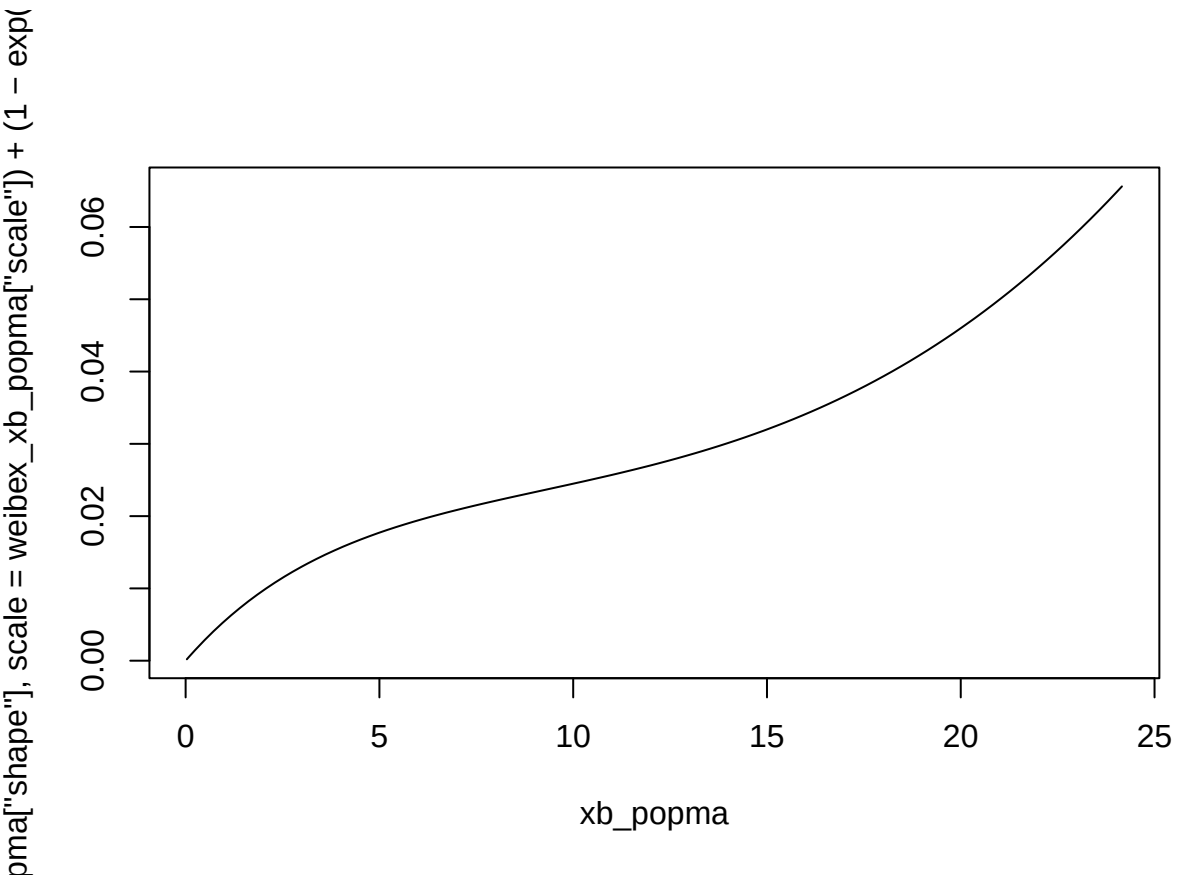



```
weibex_xb_popma
```

```
##      shape      scale       rate        mix
##  3.4449757 60.1381595  0.2532234  3.6880388
```

```
plot(xb_popma,
     (exp(weibex_xb_popma["mix"]) / ( 1 + exp(weibex_xb_popma["mix"])) *
        stats::pweibull(q = xb_popma,
```

```
                        shape = weibex_xb_popma["shape"],
                        scale = weibex_xb_popma["scale"]) +
       (1 - exp(weibex_xb_popma["mix"]) / ( 1 + exp(weibex_xb_popma["mix"]))) *
       stats::pexp(q = xb_popma,
                   rate = weibex_xb_popma["rate"])),
     type = "l")
```



Mischung von exponentiierter Weibull- und Exponentialverteilung

```
# Mischung von exponentiierter Weibull- und Exponentialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/get2weibex.R")

best_weibex2_xr_popma <- find_best_start_4parameter(p = yr_popma/100,
                                                    q = xr_popma,
                                                    max_shape = 1,
                                                    max_scale = 100,
                                                    max_rate = 0.7,
                                                    max_mix = 1,
                                                    steps_shape = 0.1,
                                                    steps_scale = 20,
                                                    steps_rate = 0.1,
                                                    steps_mix = 0.1,
                                                    fitting_function = get2weibex)
```

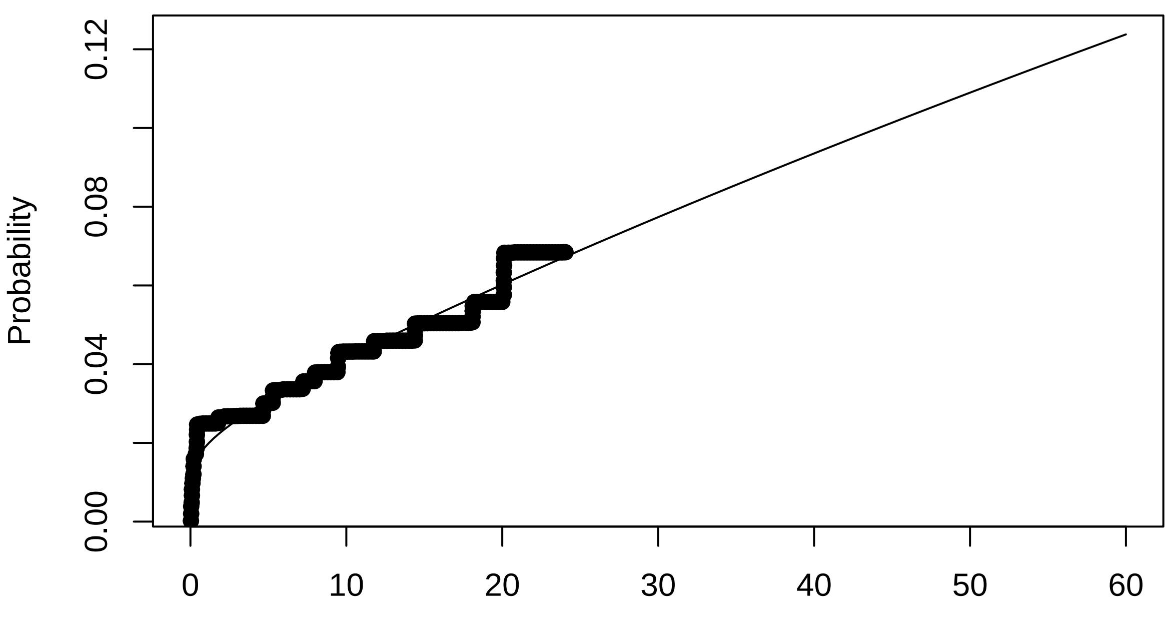
```
## Bester Startparameter: 0.5 0 0 0
```

```
## Bester Fehler: 7.134502e-08
```

```r
weibex2_xr_popma <- get2weibex(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_weibex2_xr_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.037310133 7.307485190 0.003358377 0.001000000
##
## $value
## [1] 7.134502e-08
##
## $counts
## function gradient
##       17       17
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

exp.Weibull&Exponetial(alpha= 0.0373, theta= 7.31, rate= 0.00336, mix=

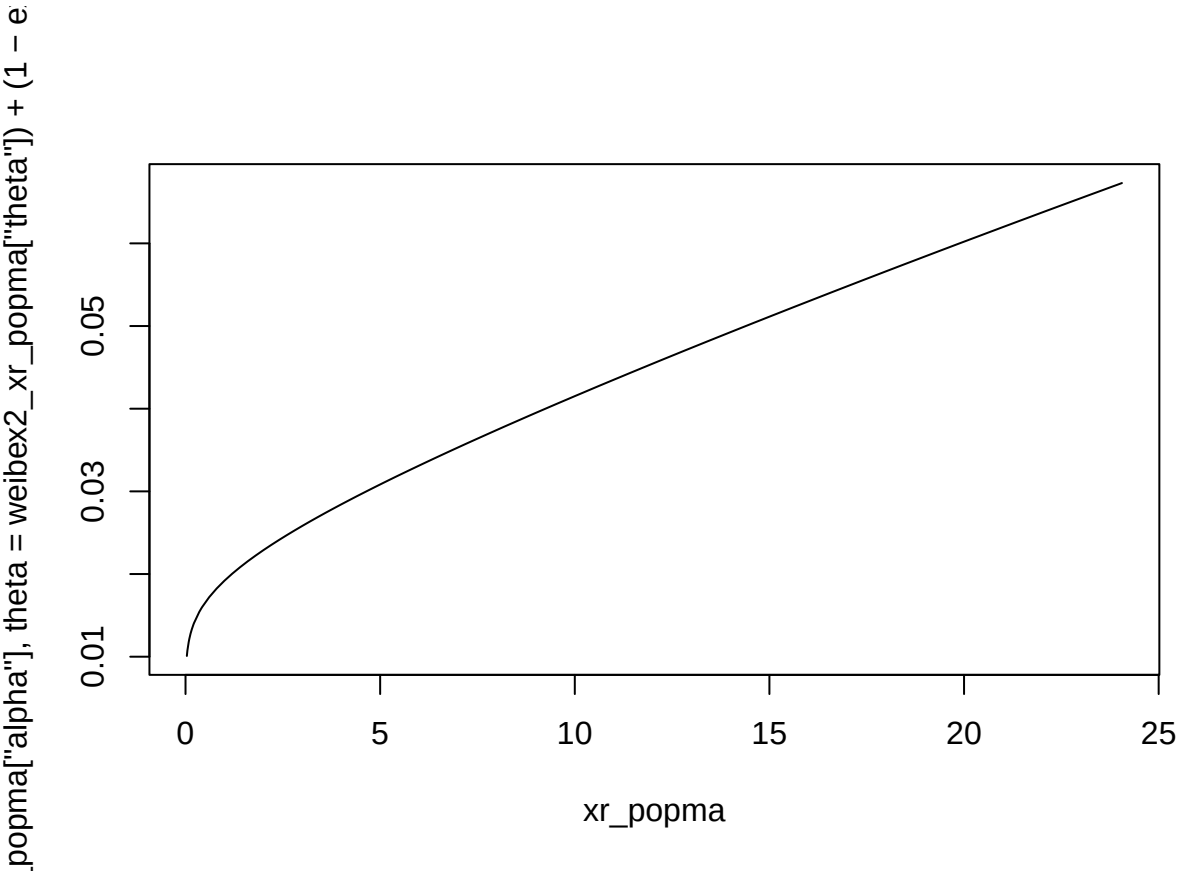


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
weibex2_xr_popma
```

```
##       alpha       theta        rate         mix
## 0.037310133 7.307485190 0.003358377 0.001000000
```

```
plot(xr_popma,
     (exp(weibex2_xr_popma["mix"]) / ( 1 + exp(weibex2_xr_popma["mix"])) *
        reliaR::pexpo.weibull(q = xr_popma,
                              alpha = weibex2_xr_popma["alpha"],
                              theta = weibex2_xr_popma["theta"]) +
        (1 - exp(weibex2_xr_popma["mix"]) / ( 1 + exp(weibex2_xr_popma["mix"]))) *
        stats::pexp(q = xr_popma,
                    rate = weibex2_xr_popma["rate"])),
     type = "l")
```

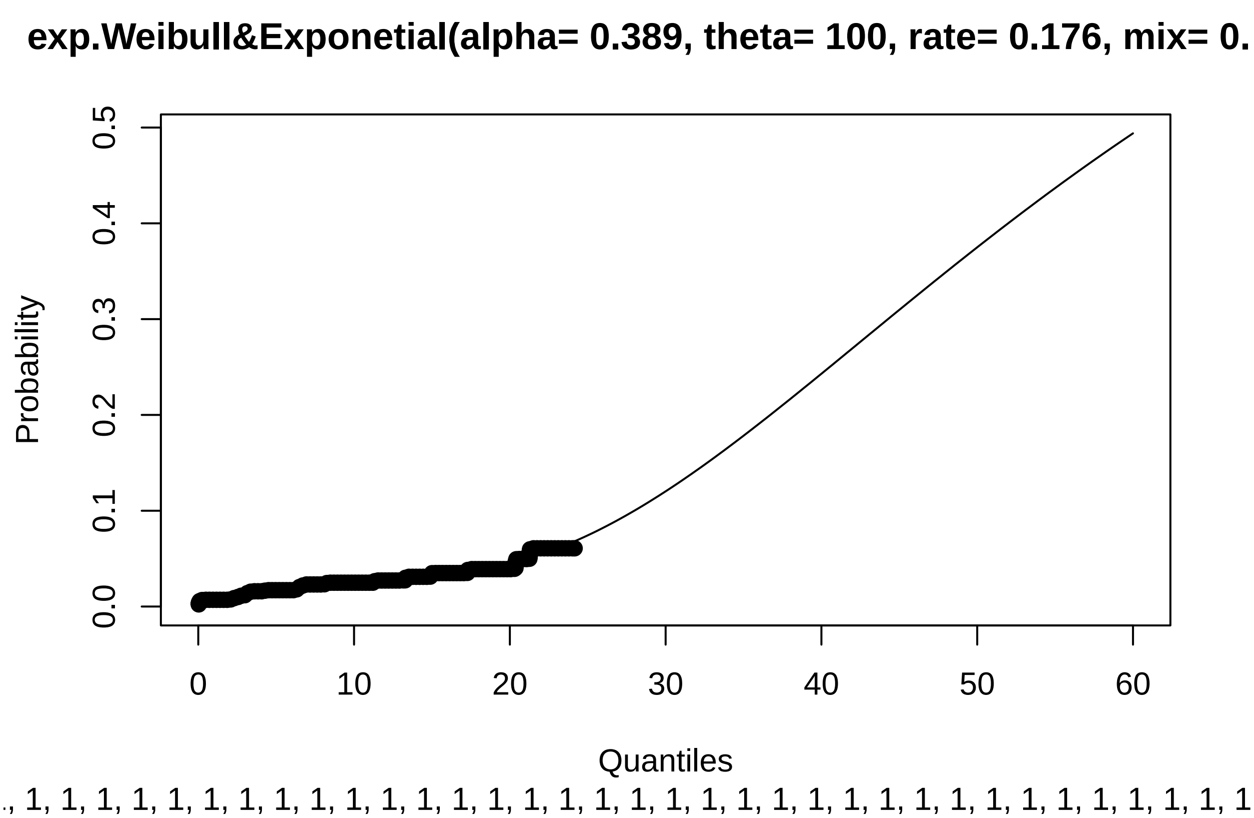



```
best_weibex2_xb_popma <- find_best_start_4parameter(p = yb_popma/100,
                                                    q = xb_popma,
                                                    max_shape = 1,
                                                    max_scale = 100,
                                                    max_rate = 0.7,
                                                    max_mix = 1,
                                                    steps_shape = 0.1,
                                                    steps_scale = 20,
                                                    steps_rate = 0.1,
                                                    steps_mix = 0.1,
                                                    fitting_function = get2weibex)
```

```
## Bester Startparameter: 0.3 100 0.7 0
## Bester Fehler: 6.483992e-08
```

```
weibex2_xb_popma <- get2weibex(
                         p = yb_popma/100,
                         q = xb_popma,
                         start = best_weibex2_xb_popma,
                         show.output = TRUE,
                         plot = TRUE
                       )
```

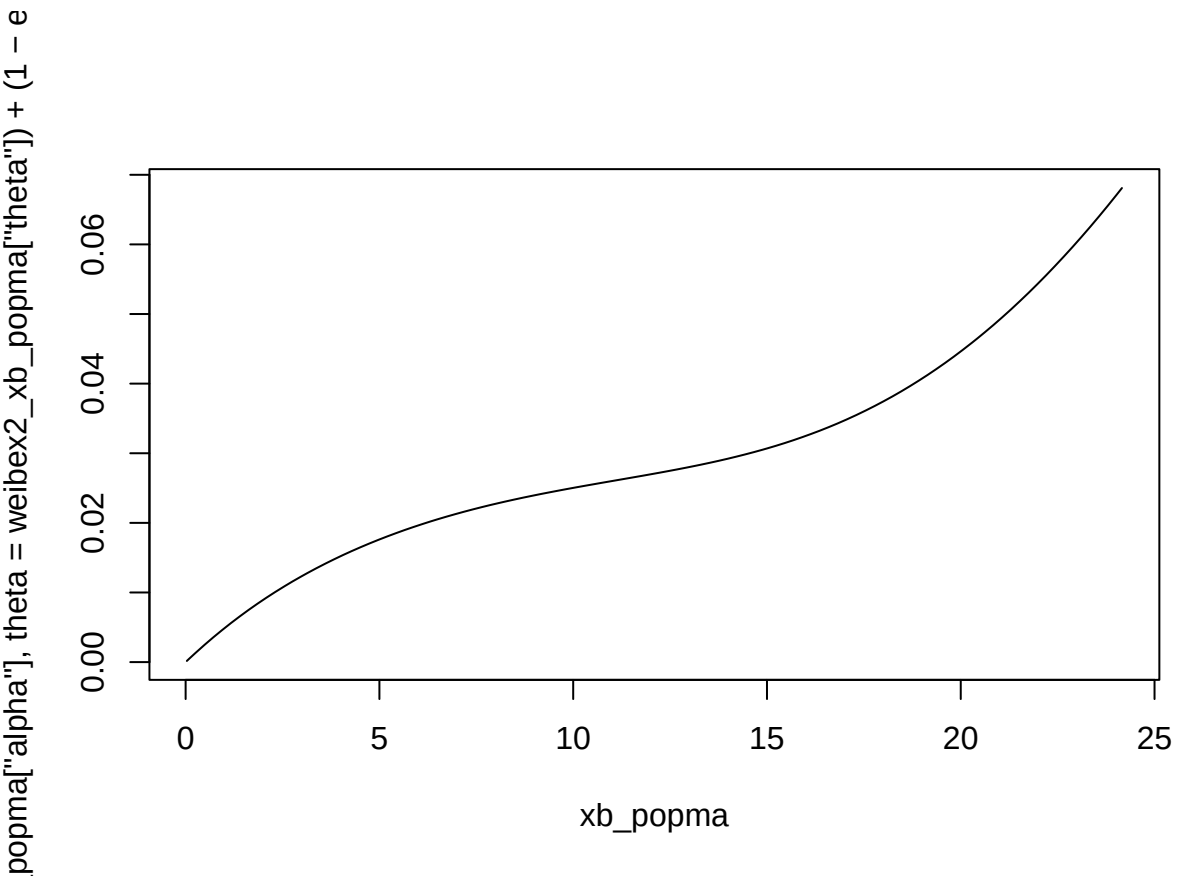
```
## $par
## [1]  0.3888109 99.9993374  0.1764586  3.4741602
##
## $value
## [1] 6.483992e-08
##
## $counts
## function gradient
##       24       24
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



```
weibex2_xb_popma
```

```
##      alpha      theta       rate        mix
##  0.3888109 99.9993374  0.1764586  3.4741602
```

```
plot(xb_popma,
     (exp(weibex2_xb_popma["mix"]) / ( 1 + exp(weibex2_xb_popma["mix"])) *
        reliaR::pexpo.weibull(q = xb_popma,
                              alpha = weibex2_xb_popma["alpha"],
                              theta = weibex2_xb_popma["theta"]) +
        (1 - exp(weibex2_xb_popma["mix"]) / ( 1 + exp(weibex2_xb_popma["mix"]))) *
        stats::pexp(q = xb_popma,
                    rate = weibex2_xb_popma["rate"])),
     type = "l")
```



Exponentiierte Weibullverteilung ohne Lambda = 1

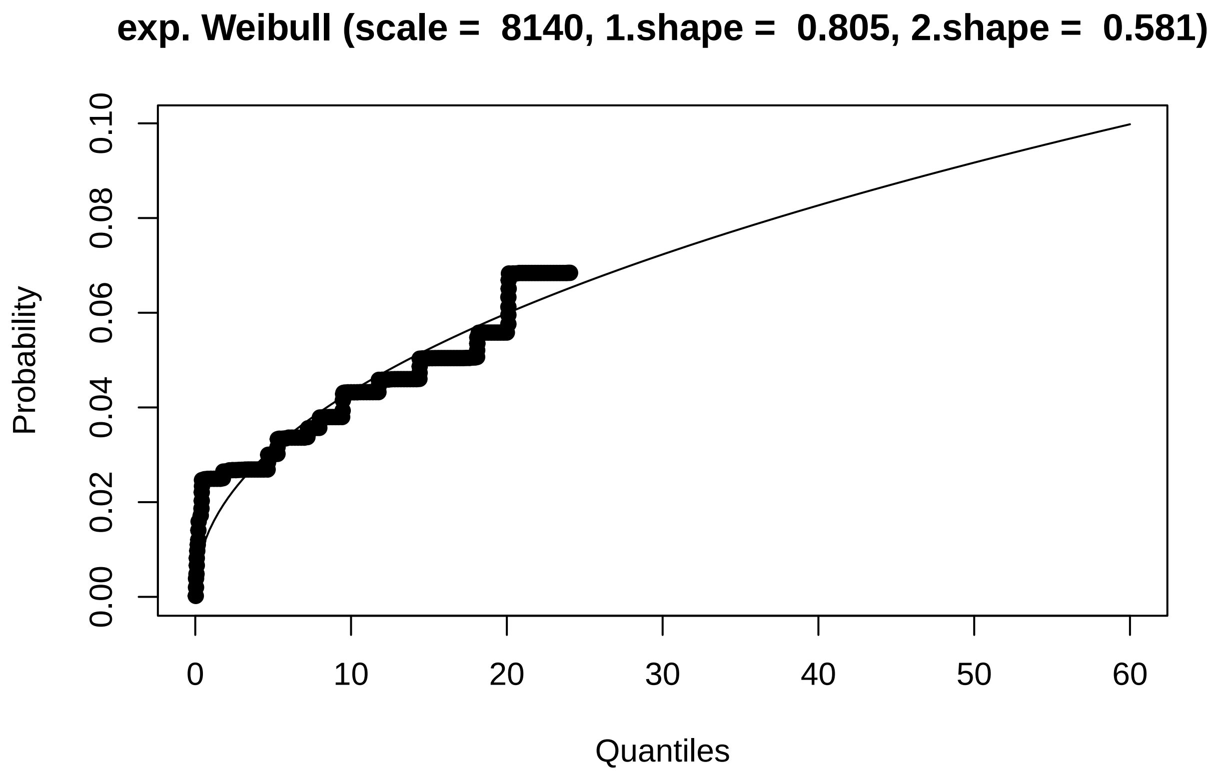
```
# exponentiierte Weibullverteilung ohne Lambda = 1
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getexpweib.R")

best_expweib_xr_popma <- find_best_start_3parameter(p = yr_popma/100,
                                                    q = xr_popma,
                                                    max_shape1 = 10,
                                                    max_shape2 = 10,
                                                    max_scale = 10,
                                                    steps_shape1 = 1,
```

```
                                                    steps_shape2 = 1,
                                                    steps_scale = 1,
                                                    fitting_function = getexpweib)
```

```
## Bester Startparameter: 1 9 4
## Bester Fehler: 1.303013e-07
```

```
expweib_xr_popma <- getexpweib(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_expweib_xr_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 8139.8360659    0.8052881    0.5814178
##
## $value
## [1] 1.303013e-07
##
## $counts
## function gradient
##      116      116
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

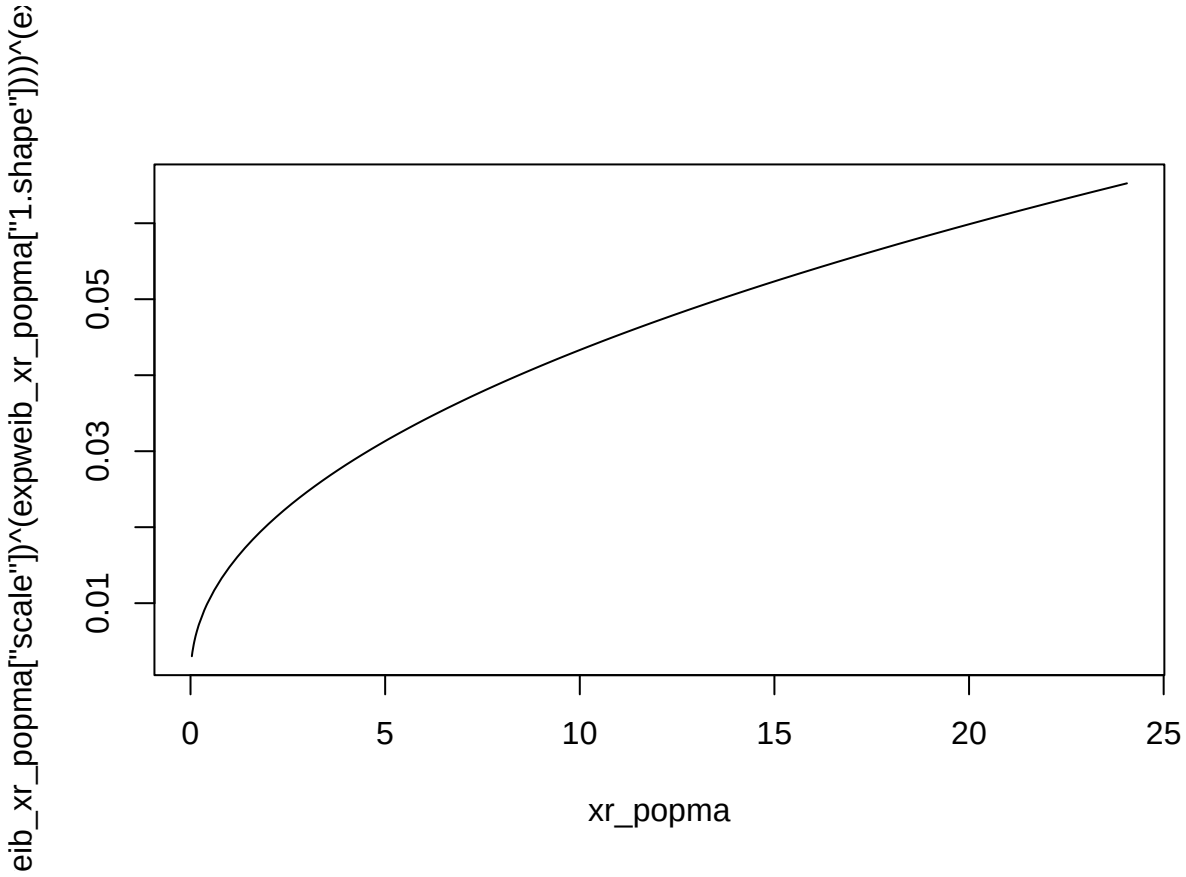


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
expweib_xr_popma
```

```
##        scale      1.shape      2.shape
## 8139.8360659    0.8052881    0.5814178
```

```
plot(xr_popma,
     (1 - exp(-(xr_popma / expweib_xr_popma["scale"])^
                     (expweib_xr_popma["1.shape"]))) ^ (expweib_xr_popma["2.shape"]),
     type = "l")
```



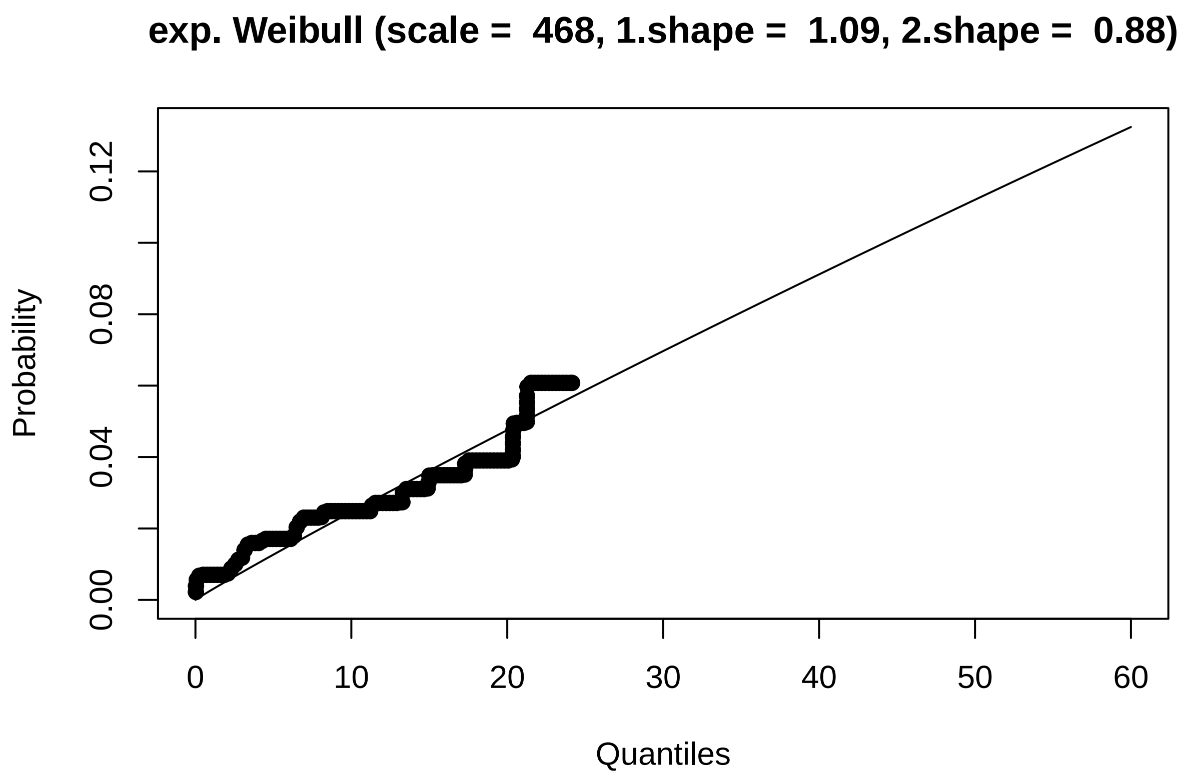

```
best_expweib_xb_popma <- find_best_start_3parameter(p = yb_popma/100,
                                                    q = xb_popma,
                                                    max_shape1 = 10,
                                                    max_shape2 = 10,
                                                    max_scale = 10,
                                                    steps_shape1 = 1,
                                                    steps_shape2 = 1,
                                                    steps_scale = 1,
                                                    fitting_function = getexpweib)
```

```
## Bester Startparameter: 5 10 3
## Bester Fehler: 1.721153e-07
```

```
expweib_xb_popma <- getexpweib(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_expweib_xb_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 468.0210001   1.0930591   0.8798307
##
## $value
## [1] 1.721153e-07
##
```

```
## $counts
## function gradient
##       90       90
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

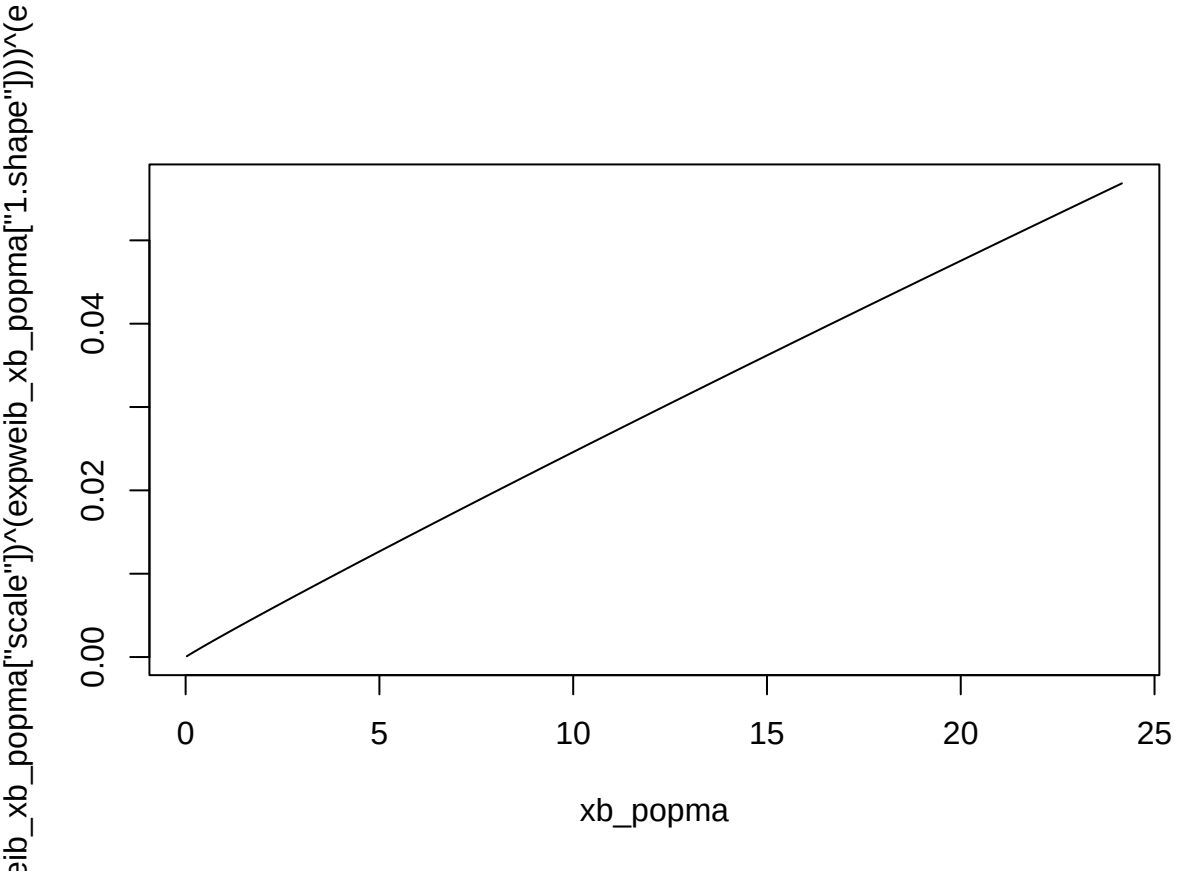


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
expweib_xb_popma
```

```
##       scale     1.shape     2.shape
## 468.0210001   1.0930591   0.8798307
```

```
plot(xb_popma,
     (1 - exp(-(xb_popma / expweib_xb_popma["scale"])^(expweib_xb_popma["1.shape"])))^(expweib_xb_popma
     type = "l")
```



3-Stufige Exponetialverteilung

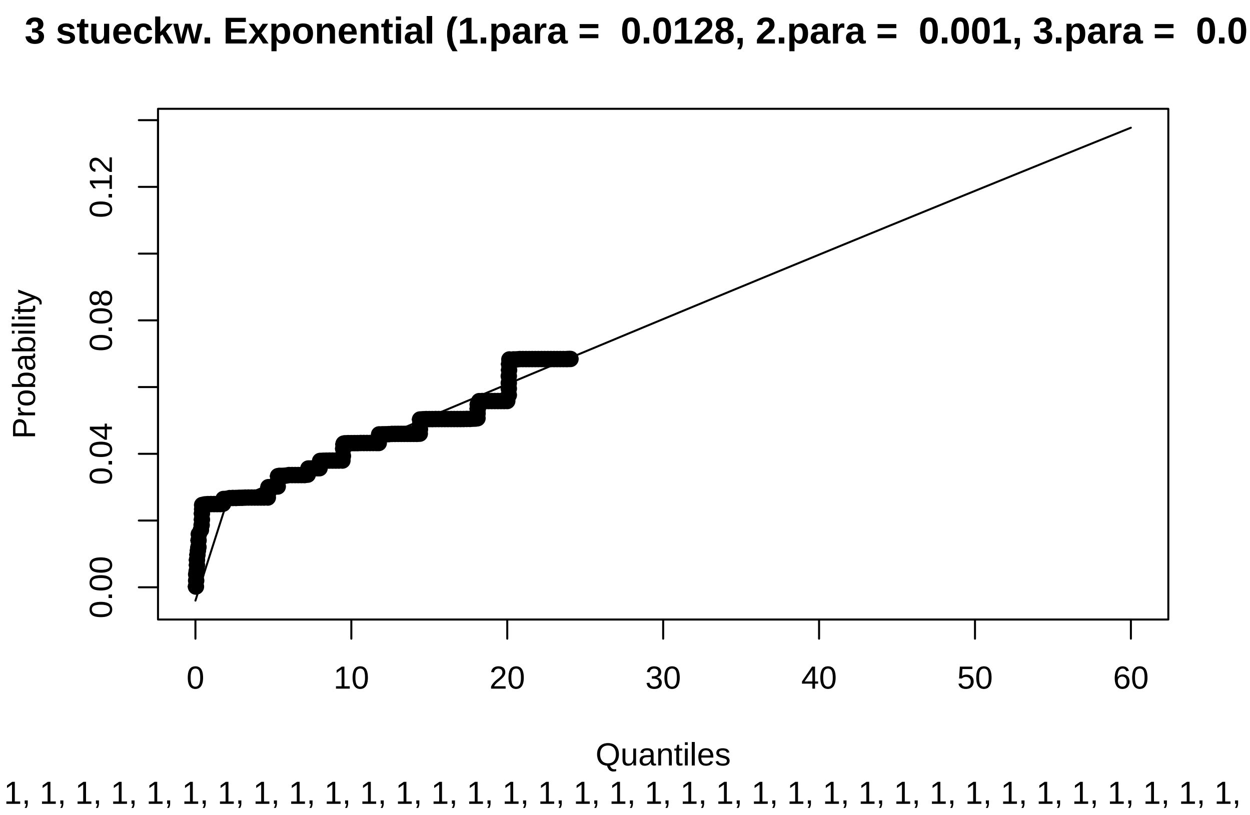
```
# 3-Stufige Exponetialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getstuexp3.R")

best_stuexp3_xr_popma <- find_best_start_3parameter(p = yr_popma/100,
                                                    q = xr_popma,
                                                    max_shape1 = 0.1,
                                                    max_shape2 = 0.1,
                                                    max_scale = 0.1,
                                                    steps_shape1 = 0.01,
                                                    steps_shape2 = 0.01,
                                                    steps_scale = 0.01,
                                                    fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0 0 0.01
## Bester Fehler: 2.110647e-07
```

```
stuexp3_xr_popma <- getstuexp3(
                         p = yr_popma/100,
                         q = xr_popma,
                         start = best_stuexp3_xr_popma,
                         show.output = TRUE,
                         plot = TRUE,
                         wert1 = 2,
                         wert2 = 6
```

```
                      )
```

```
## $par
## [1] 0.01279132 0.00100000 0.00100000
##
## $value
## [1] 2.110647e-07
##
## $counts
## function gradient
##       42       42
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

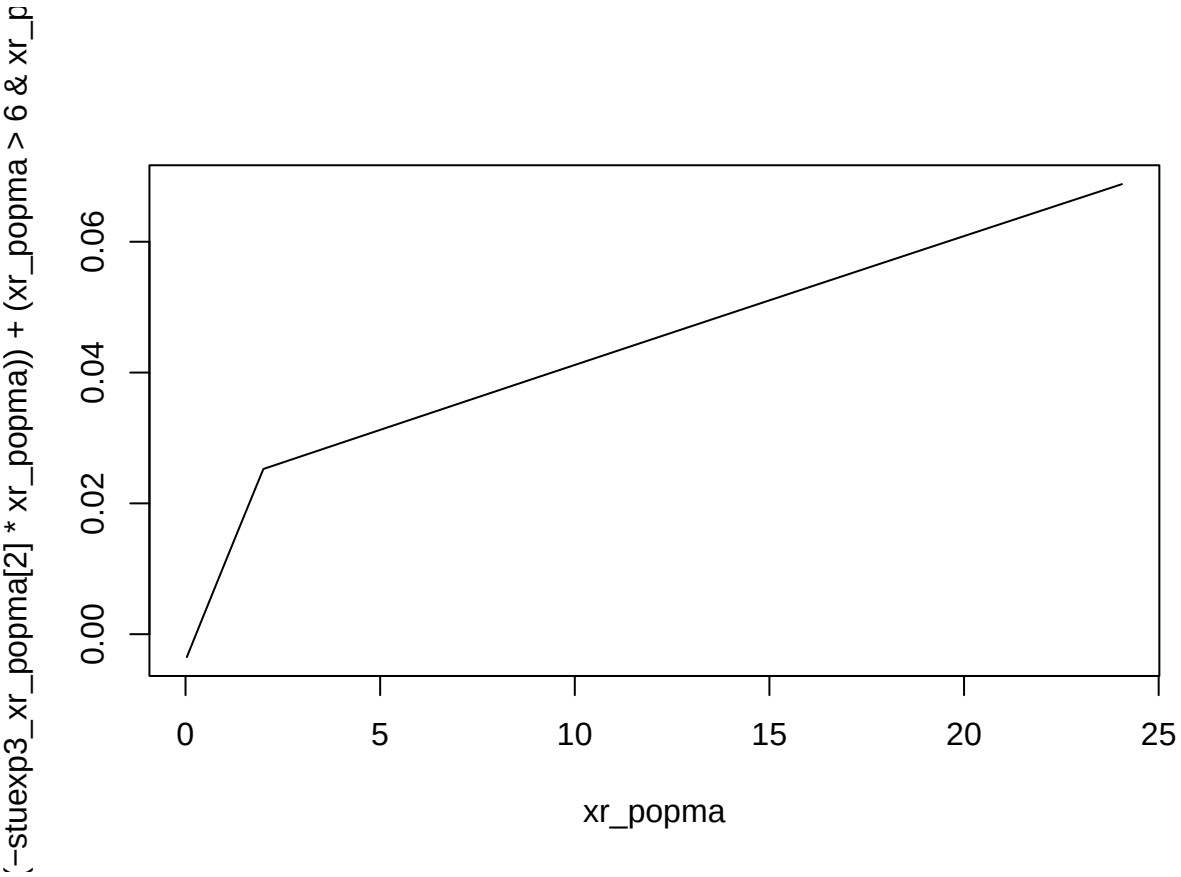



```
stuexp3_xr_popma
```

```
##     1.para     2.para     3.para
## 0.01279132 0.00100000 0.00100000
```

```
plot(xr_popma,
     ((xr_popma > 0 & xr_popma <= 2 ) * (1 - exp(-stuexp3_xr_popma[1] * xr_popma)) +
                              (xr_popma > 2 & xr_popma <= 6 ) *
              (1 - exp(-stuexp3_xr_popma[1] * 2))  +
```

```
                                    (exp(-2 * stuexp3_xr_popma[2]) -
                                            exp(-stuexp3_xr_popma[2] * xr_popma)) +
                                    (xr_popma > 6 & xr_popma <= 65 ) *
            (1 - exp(-stuexp3_xr_popma[1] * 2)) +
                                    (exp(-2 * stuexp3_xr_popma[2]) - exp(-6 * stuexp3_xr_popma[2])) +
                                    (exp(-6 * stuexp3_xr_popma[3]) -
                                            exp(-stuexp3_xr_popma[3] * xr_popma))),
    type = "l")
```



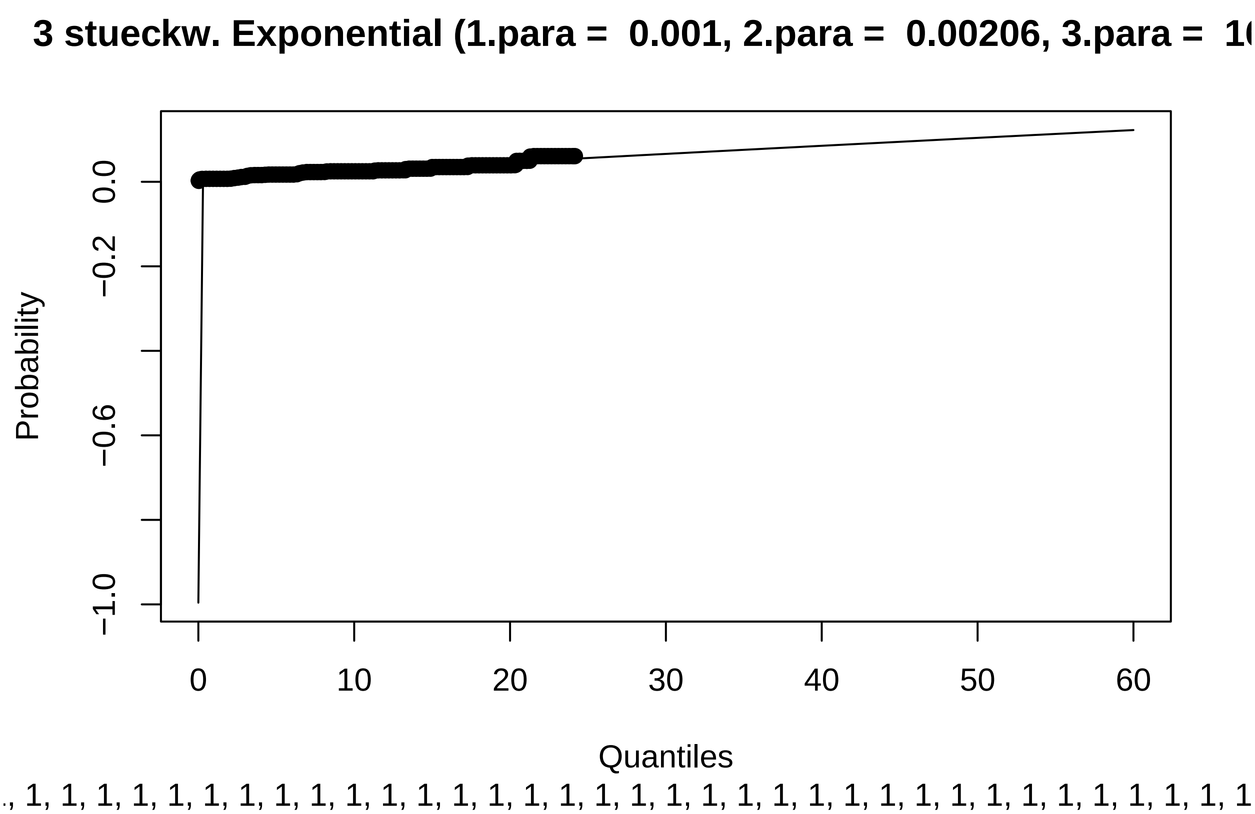

```
best_stuexp3_xb_popma <- find_best_start_3parameter(p = yb_popma/100,
                                                    q = xb_popma,
                                                    max_shape1 = 0.1,
                                                    max_shape2 = 0.1,
                                                    max_scale = 0.1,
                                                    steps_shape1 = 0.01,
                                                    steps_shape2 = 0.01,
                                                    steps_scale = 0.01,
                                                    fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0.08 0.08 0.1
## Bester Fehler: 1.538679e-07
```

```
stuexp3_xb_popma <- getstuexp3(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_stuexp3_xb_popma,
                        show.output = TRUE,
```

```
                        plot = TRUE,
                        wert1 = 2,
                        wert2 = 6
                      )
```

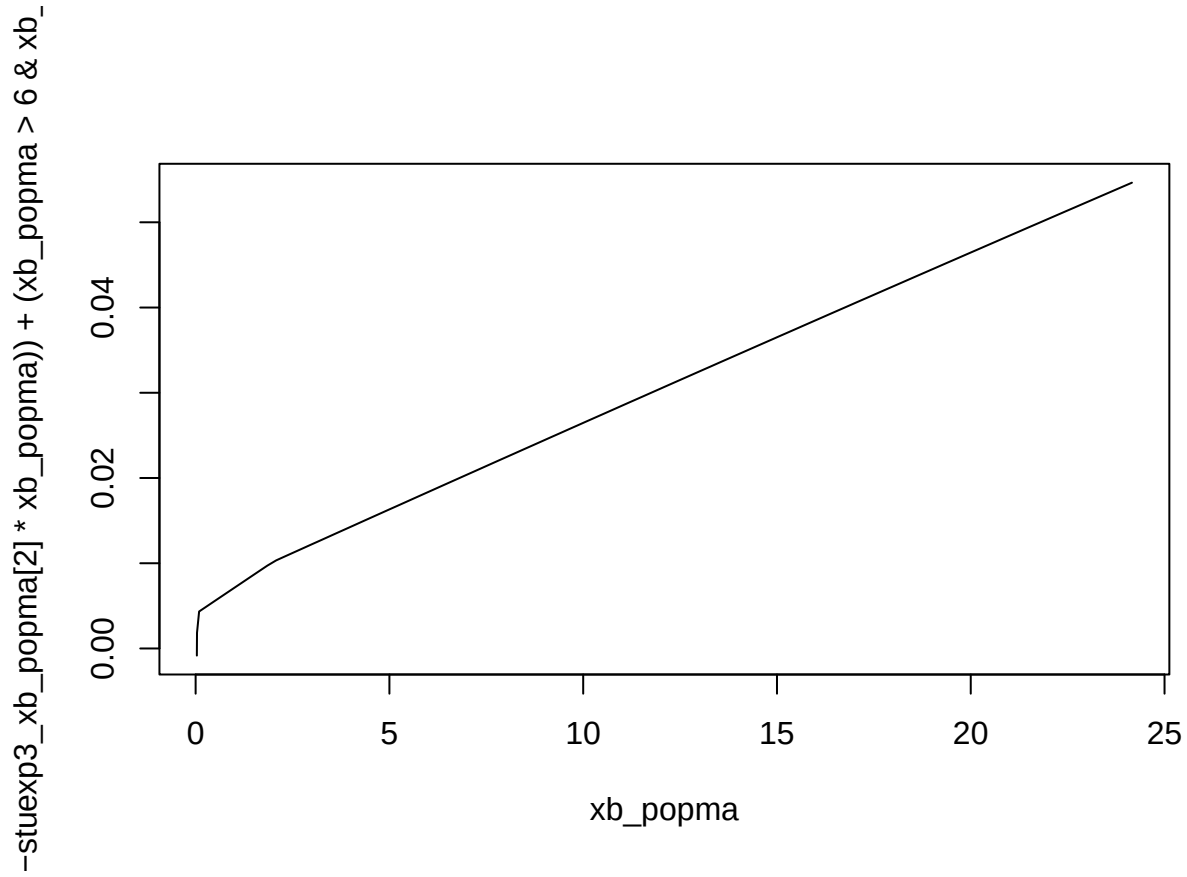
```
## $par
## [1] 1.000000e-03 2.061478e-03 1.690237e+02
##
## $value
## [1] 1.538679e-07
##
## $counts
## function gradient
##       45       45
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```




```
stuexp3_xb_popma
```

```
##       1.para       2.para       3.para
## 1.000000e-03 2.061478e-03 1.690237e+02
```

```r
plot(xb_popma,
     ((xb_popma > 0 & xb_popma <= 2 ) * (1 - exp(-stuexp3_xb_popma[1] * xb_popma)) +
                                  (xb_popma > 2 & xb_popma <= 6 ) *
              (1 - exp(-stuexp3_xb_popma[1] * 2))  +
                                  (exp(-2 * stuexp3_xb_popma[2]) - exp(-stuexp3_xb_popma[2] *
                                                                       xb_popma)) +
                                  (xb_popma > 6 & xb_popma <= 65 ) *
              (1 - exp(-stuexp3_xb_popma[1] * 2)) +
                                  (exp(-2 * stuexp3_xb_popma[2]) - exp(-6 * stuexp3_xb_popma[2])) +
                                  (exp(-6 * stuexp3_xb_popma[3]) -
                                           exp(-stuexp3_xb_popma[3] * xb_popma))),
     type = "l")
```



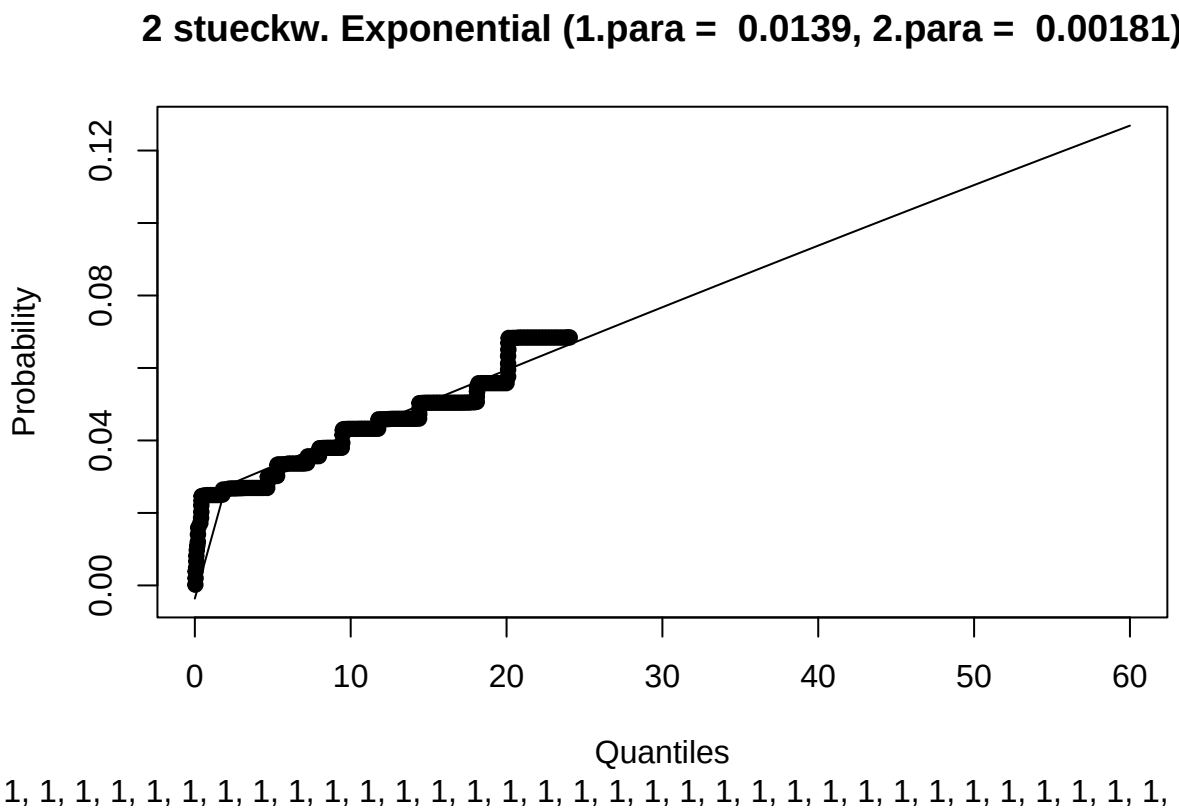
2-Stufige Exponetialverteilung

```r
# 2-Stufige Exponetialverteilung
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getstuexp2.R")

best_stuexp2_xr_popma <- find_best_start_2parameter(p = yr_popma/100,
                                                    q = xr_popma,
                                                    max_beta = 1,
                                                    max_eta = 0.5,
                                                    steps_beta = 0.1,
                                                    steps_eta = 0.01,
                                                    fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0.1 0.1
## Bester Fehler: 2.052669e-07
```

```
stuexp2_xr_popma <- getstuexp2(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_stuexp2_xr_popma,
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2
                      )
```

```
## $par
## [1] 0.013928893 0.001811953
##
## $value
## [1] 2.052669e-07
##
## $counts
## function gradient
##       45       45
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

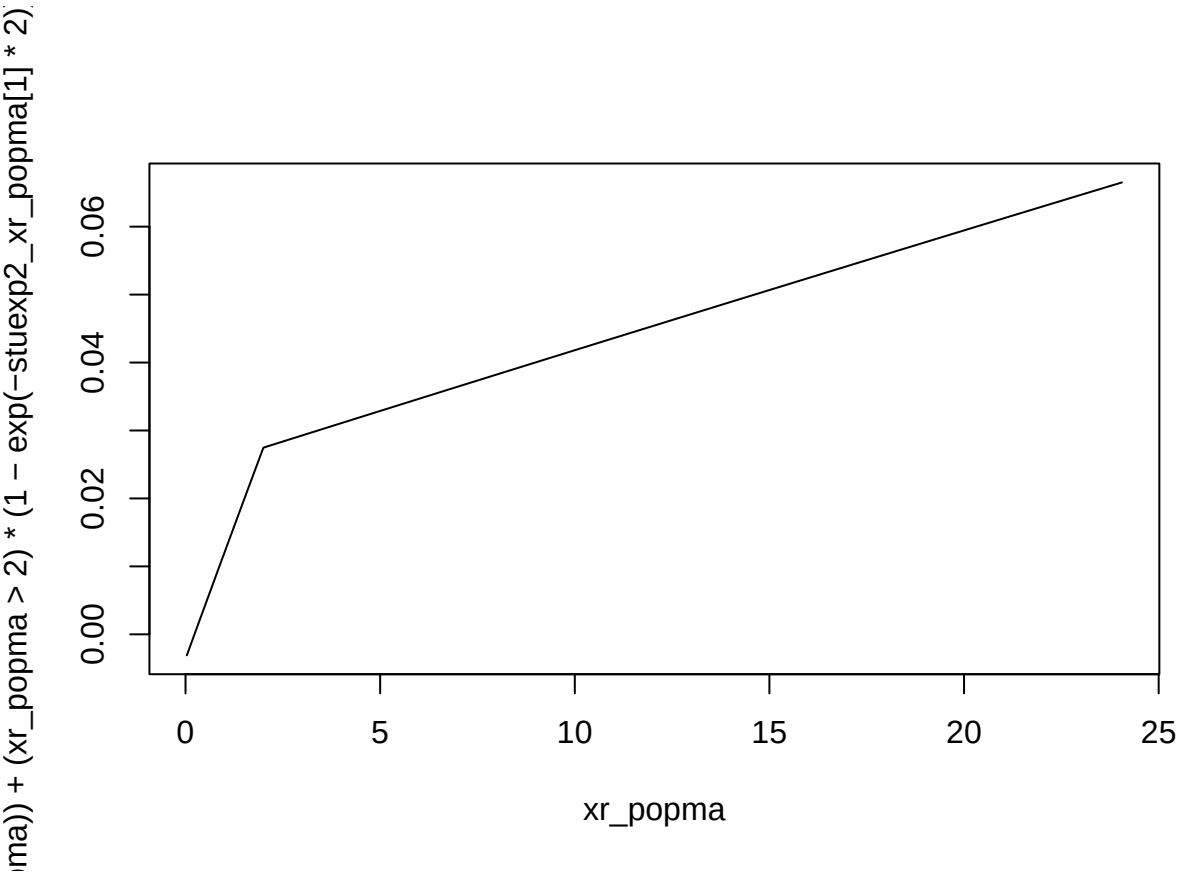


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
stuexp2_xr_popma
```

```
##      1.para      2.para
## 0.013928893 0.001811953
```

```
plot(xr_popma,
     ((xr_popma > 0 & xr_popma <= 2 ) * (1 - exp(-stuexp2_xr_popma[1] * xr_popma)) +
                                      (xr_popma > 2 ) *  (1 - exp(-stuexp2_xr_popma[1] * 2))  +
                                      (exp(-2 * stuexp2_xr_popma[2]) -
                                               exp(-stuexp2_xr_popma[2] * xr_popma))),
     type = "l")
```



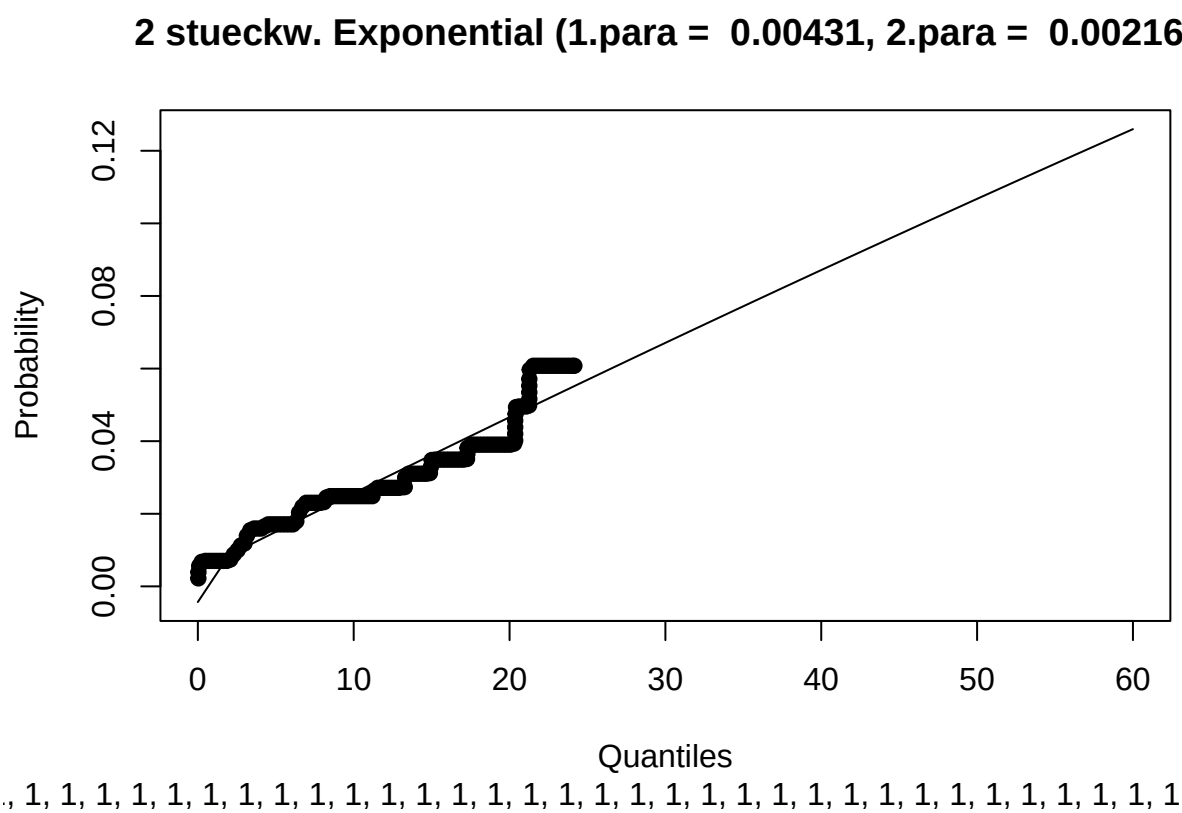

```
best_stuexp2_xb_popma <- find_best_start_2parameter(p = yb_popma/100,
                                                    q = xb_popma,
                                                    max_beta = 1,
                                                    max_eta = 0.5,
                                                    steps_beta = 0.1,
                                                    steps_eta = 0.01,
                                                    fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0 0.02
## Bester Fehler: 1.730793e-07
```

```
stuexp2_xb_popma <- getstuexp2(
                        p = yb_popma/100,
                        q = xb_popma,
                        start = best_stuexp2_xb_popma,
```

```
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2
                      )
```

```
## $par
## [1] 0.004312672 0.002162521
##
## $value
## [1] 1.730793e-07
##
## $counts
## function gradient
##       16       16
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

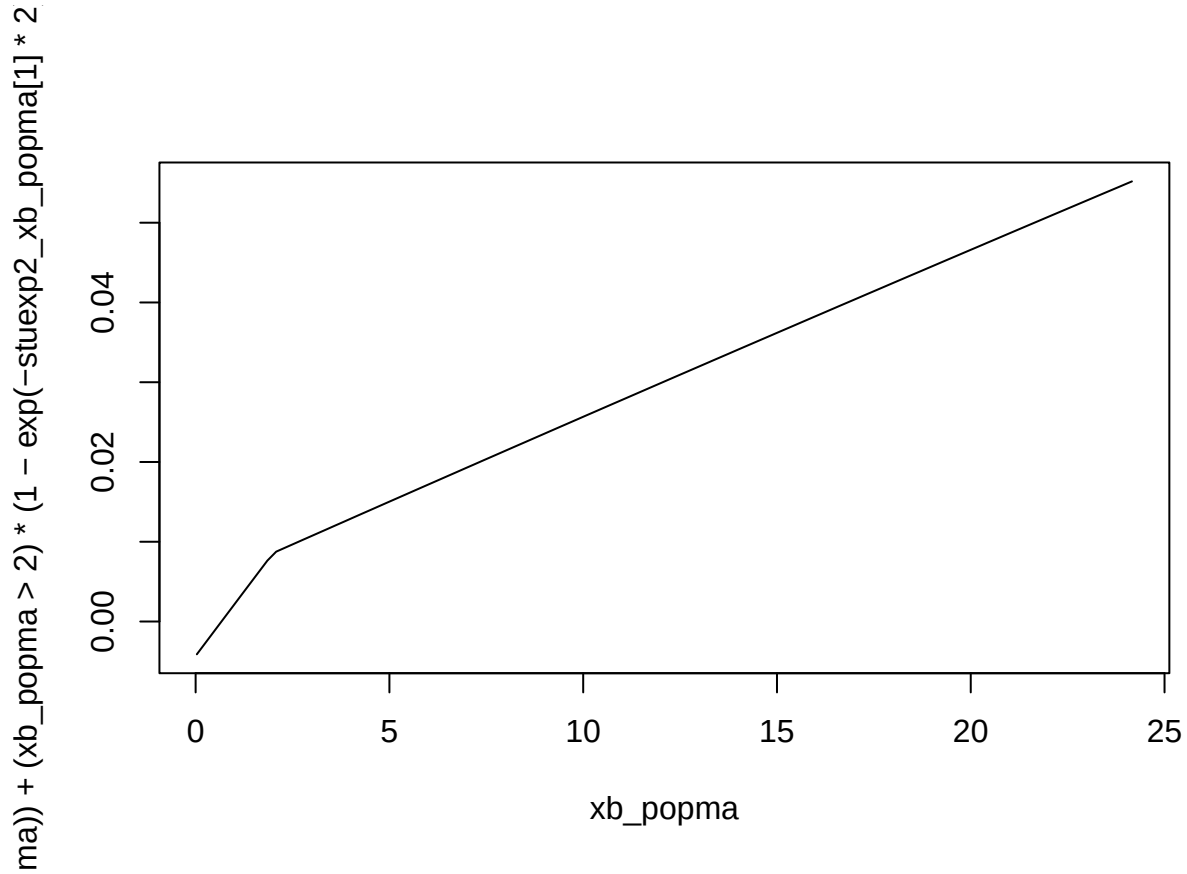


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
stuexp2_xb_popma
```

```
##      1.para      2.para
## 0.004312672 0.002162521
```

```r
plot(xb_popma,
     ((xb_popma > 0 & xb_popma <= 2 ) * (1 - exp(-stuexp2_xb_popma[1] * xb_popma)) +
                                      (xb_popma > 2 ) *  (1 - exp(-stuexp2_xb_popma[1] * 2))  +
                                      (exp(-2 * stuexp2_xb_popma[2]) -
                                              exp(-stuexp2_xb_popma[2] * xb_popma))),
     type = "l")
```



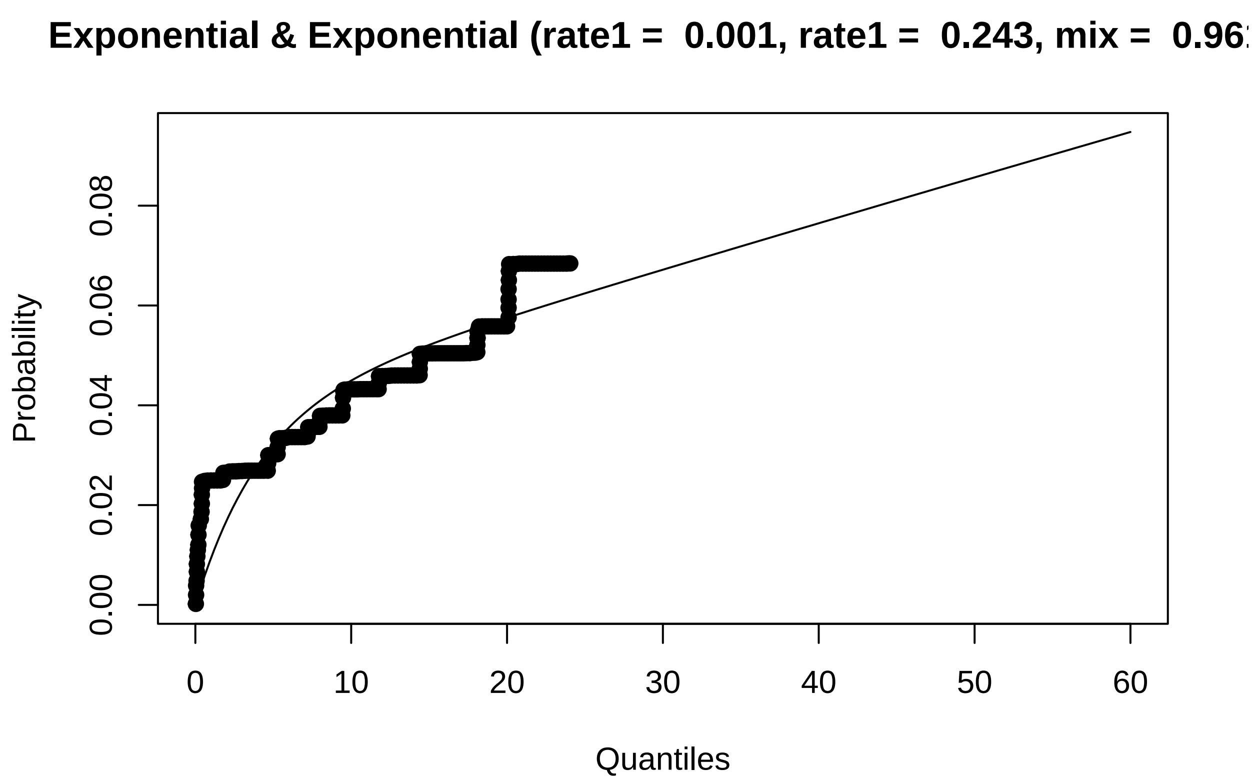
Mischung aus 2 Exponentialverteilungen

```r
# Mischung aus 2 Exponentialverteilungen
source("C:/Users/nhonh/OneDrive/Dokumente/Unikrams/Masterarbeit/R Funktionen/getexex.R")

best_exex_xr_popma <- find_best_start_3parameter(p = yr_popma/100,
                                                 q = xr_popma,
                                                 max_shape1 = 0.7,
                                                 max_shape2 = 0.7,
                                                 max_scale = 1,
                                                 steps_shape1 = 0.1,
                                                 steps_shape2 = 0.1,
                                                 steps_scale = 0.1,
                                                 fitting_function = getexex)
```

```
## Bester Startparameter: 0.1 0.6 0.7
## Bester Fehler: 2.667073e-07
```

```r
exex_xr_popma <- getexex(
                        p = yr_popma/100,
                        q = xr_popma,
                        start = best_exex_xr_popma,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.0010000 0.2431941 3.2100466
##
## $value
## [1] 2.667073e-07
##
## $counts
## function gradient
##       18       18
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



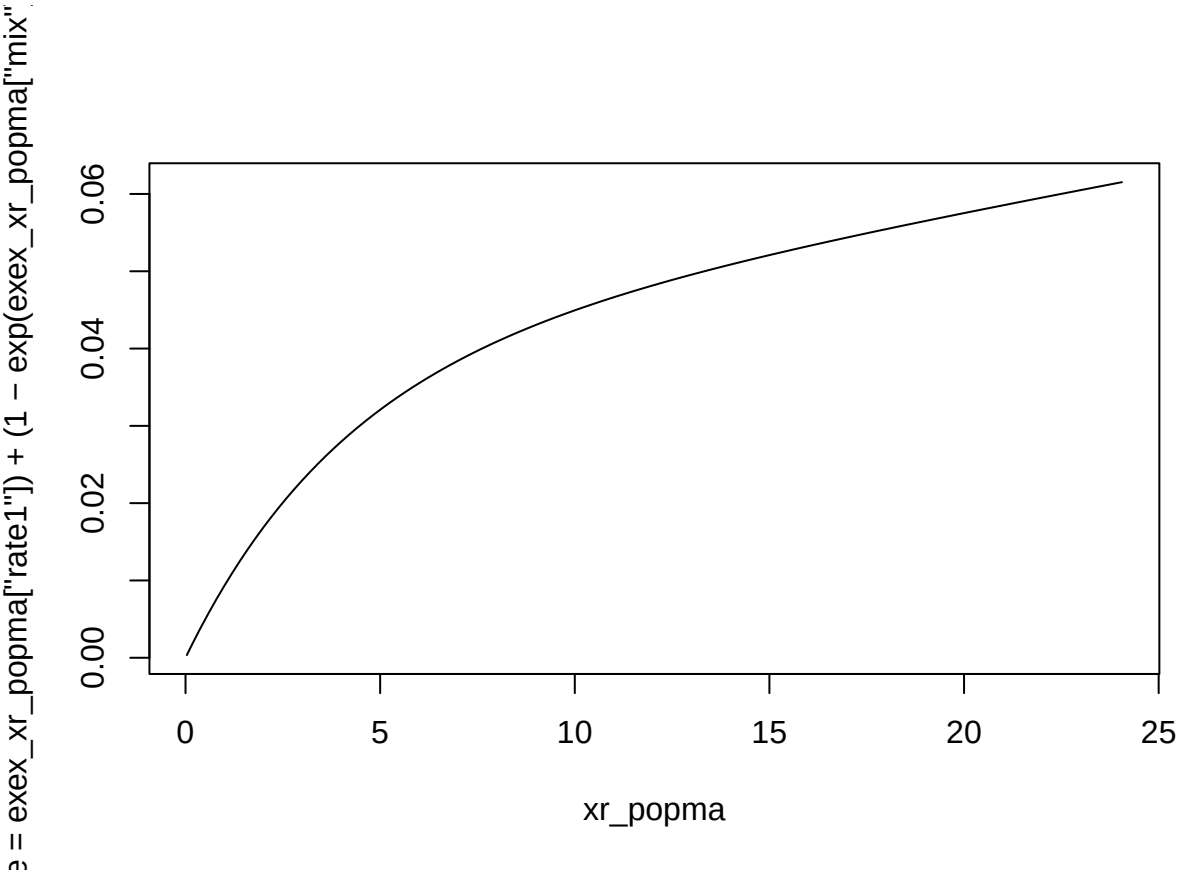
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```r
exex_xr_popma
```

```
##      rate1      rate2       mix
```

```
## 0.0010000 0.2431941 3.2100466
```

```
plot(xr_popma,
     (exp(exex_xr_popma["mix"]) / ( 1 + exp(exex_xr_popma["mix"])) *  stats::pexp(q = xr_popma,
                                                                                  rate = exex_xr_popma["rate1
                        (1 - exp(exex_xr_popma["mix"]) / ( 1 + exp(exex_xr_popma["mix"]))) *
               stats::pexp(q = xr_popma,
                           rate = exex_xr_popma["rate2"])),
     type = "l")
```



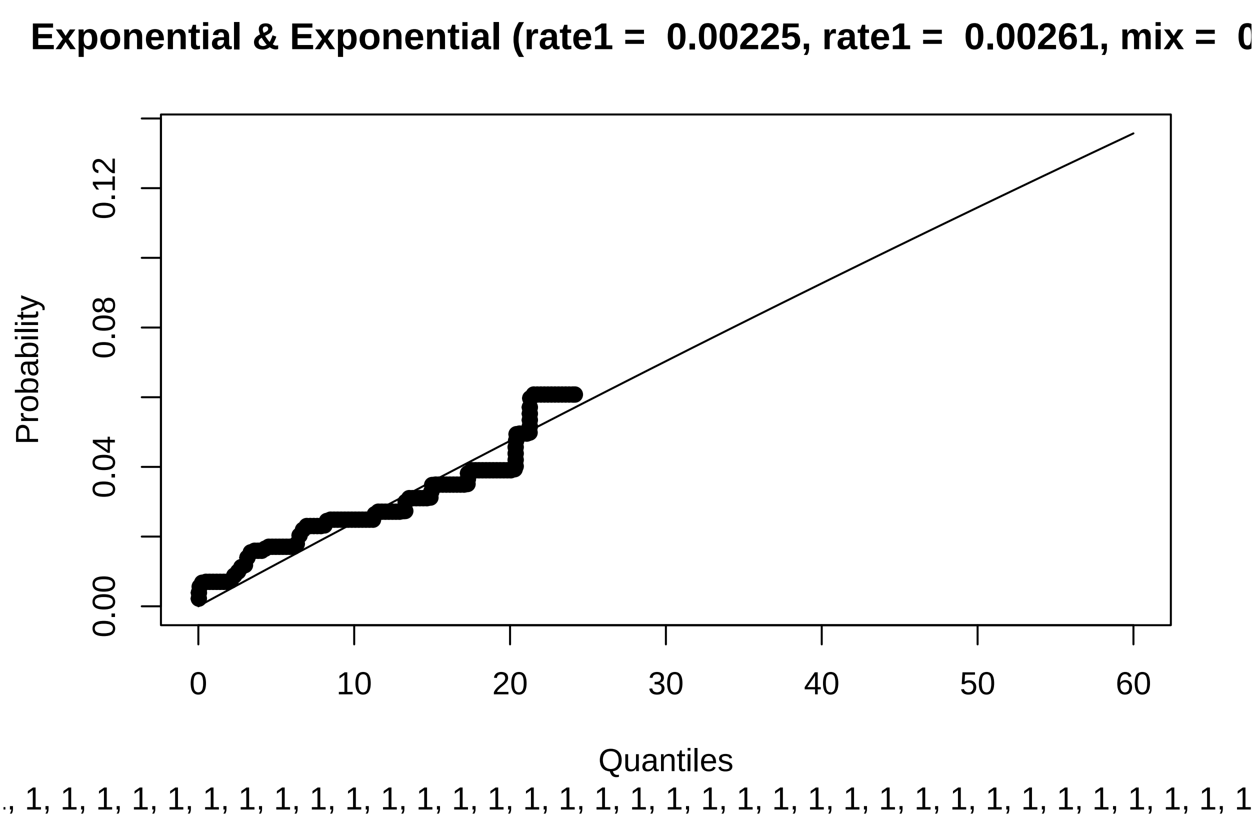

```
best_exex_xb_popma <- find_best_start_3parameter(p = yb_popma/100,
                                                 q = xb_popma,
                                                 max_shape1 = 0.7,
                                                 max_shape2 = 0.7,
                                                 max_scale = 1,
                                                 steps_shape1 = 0.1,
                                                 steps_shape2 = 0.1,
                                                 steps_scale = 0.1,
                                                 fitting_function = getexex)
```

```
## Bester Startparameter: 0.3 0.3 0
## Bester Fehler: 1.761187e-07
```

```
exex_xb_popma <- getexex(
                         p = yb_popma/100,
                         q = xb_popma,
                         start = best_exex_xb_popma,
                         show.output = TRUE,
```

```
                           plot = TRUE
                         )
```

```
## $par
## [1] 0.002252595 0.002611439 0.001000346
##
## $value
## [1] 1.761187e-07
##
## $counts
## function gradient
##       11       11
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

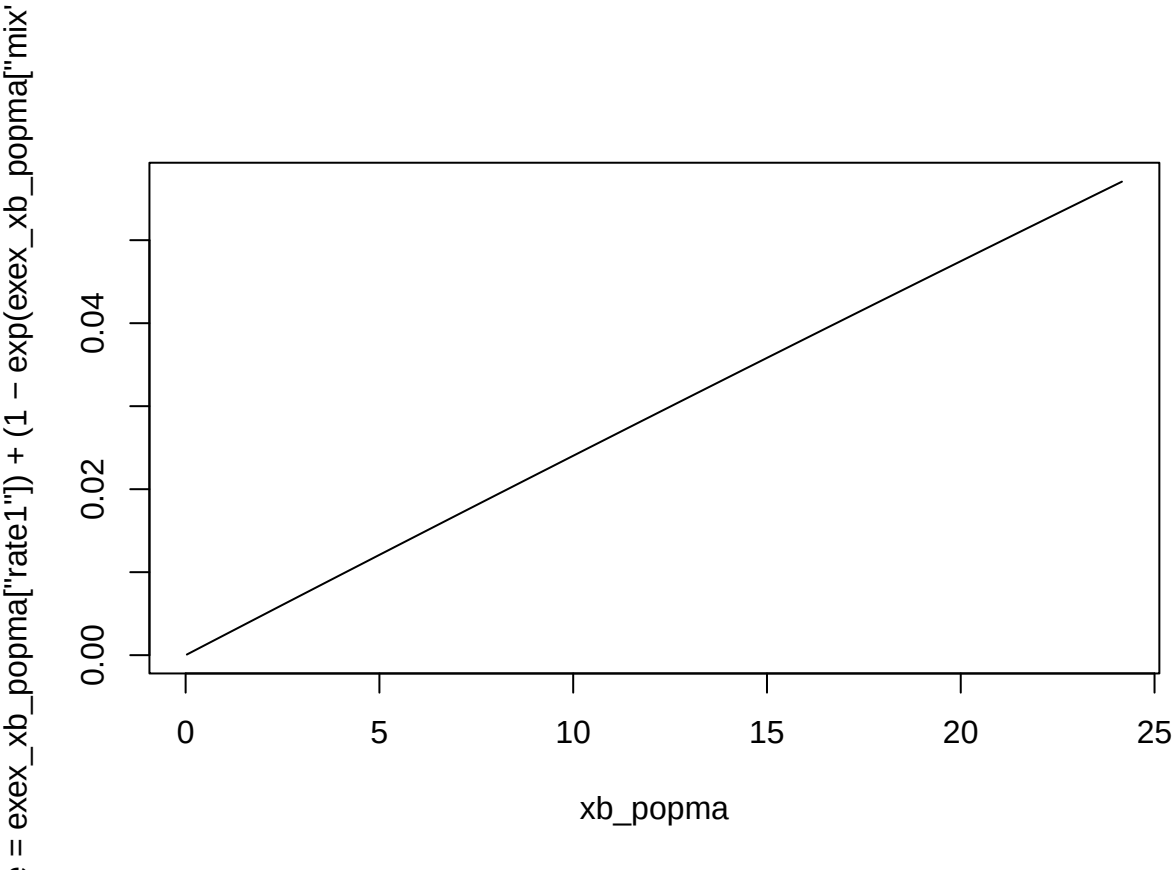



```
exex_xb_popma
```

```
##       rate1       rate2         mix
## 0.002252595 0.002611439 0.001000346
```

```
plot(xb_popma,
     (exp(exex_xb_popma["mix"]) / ( 1 + exp(exex_xb_popma["mix"])) *  stats::pexp(q = xb_popma,
                                                                              rate = exex_xb_popma["rate1
```

```
                    (1 - exp(exex_xb_popma["mix"]) / ( 1 + exp(exex_xb_popma["mix"]))) *
            stats::pexp(q = xb_popma,
                        rate = exex_xb_popma["rate2"])),
    type = "l")
```



Ergebnnis

```
getvalue <- function(p, q, best_start, fitting_function){
  if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
    output <- capture.output({
    result <- getstuexp2(p = p, q = q, start = best_start, show.output = TRUE,
                         plot = FALSE, wert1 = 2)
    })
  }
  else if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                          p = p, q = q, start = best_start,
                          show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
  }
  else{
    output <- capture.output({
    result <- fitting_function(p = p, q = q, start = best_start,
```

```
                                show.output = TRUE, plot = FALSE)
    })
  }

  # Berechne den Fehler (popma_r_1$value) für die aktuellen Startparameter
  error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

  return(error)
}

best_test <- function(p, q, weibull, weib, weibex, weibex2, expweib, stuexp3, stuexp2,
                      exex, start_weibull, start_weib, start_weibex, start_weibex2,
                      start_expweib, start_stuexp3, start_stuexp2, start_exex,
                      group){
  weibull_val <- getvalue(p, q, start_weibull, getweibullpar)
  weib_val <- getvalue(p, q, start_weib, getweibpar)
  weibex_val <- getvalue(p, q, start_weibex, getweibex)
  weibex2_val <- getvalue(p, q, start_weibex2, get2weibex)
  expweib_val <- getvalue(p, q, start_expweib, getexpweib)
  stuexp3_val <- getvalue(p, q, start_stuexp3, getstuexp3)
  stuexp2_val <- getvalue(p, q, start_stuexp2, getstuexp2)
  exex_val <- getvalue(p, q, start_exex, getexex)

  error_distribution_pairs <- list(
    list(weibull_val, "W"),
    list(weib_val, "e.W."),
    list(weibex_val, "M. W&E"),
    list(weibex2_val, "M. e.W&E"),
    list(expweib_val, "e.W o. lambda = 1"),
    list(stuexp3_val, "3 s.E."),
    list(stuexp2_val, "2 s.E."),
    list(exex_val, "M. E&E")
  )

  # Suchen Verteilung mit dem kleinsten Fehler
  best_pair <- error_distribution_pairs[[which.min(sapply(
    error_distribution_pairs, function(pair) pair[[1]]))]]

  # Drucken Sie die Ergebnisse
  cat("Beste Verteilung:", best_pair[[2]], "\n")
  cat("Bester Fehler:", best_pair[[1]], "\n")
  cat("Gruppe: ", group)

  return(c(group, best_pair[[2]],  best_pair[[1]]))
}

best_tavr <- best_test(yb_popma/100, xb_popma, popma_b_1, weibbull_xb_popma,
                       weibex_xb_popma, weibex2_xb_popma, expweib_xb_popma,
                       stuexp3_xb_popma, stuexp2_xb_popma, exex_xb_popma,
                       best_popma_b_1, best_weibbull_xb_popma,
                       best_weibex_xb_popma, best_weibex2_xb_popma,
                       best_expweib_xb_popma, best_stuexp3_xb_popma,
                       best_stuexp2_xb_popma, best_exex_xb_popma, "TAVR")
```

```
## Beste Verteilung: M. W&E
## Bester Fehler: 6.267981e-08
## Gruppe:  TAVR
```

```
best_savr <- best_test(yr_popma/100, xr_popma, popma_r_1, weibbull_xr_popma,
                       weibex_xr_popma, weibex2_xr_popma, expweib_xr_popma,
                       stuexp3_xr_popma, stuexp2_xr_popma, exex_xr_popma,
                       best_popma_r_1, best_weibbull_xr_popma,
                       best_weibex_xr_popma, best_weibex2_xr_popma,
                       best_expweib_xr_popma, best_stuexp3_xr_popma,
                       best_stuexp2_xr_popma, best_exex_xr_popma, "SAVR")
```

```
## Beste Verteilung: M. W&E
## Bester Fehler: 6.242942e-08
## Gruppe:  SAVR
```

```
tab <- matrix(c("Medtronic", "NiRi", "TSmF", best_tavr[1], best_tavr[2],
                best_tavr[3], NA, NA, weibex_xb_popma[1:2], NA,
                weibex_xb_popma[3:4],
                "Medtronic", "NiRi", "TSmF",  best_savr[1], best_savr[2],
                best_savr[3], NA, NA, weibex_xr_popma[1:2], NA,
                weibex_xr_popma[3:4]),
              ncol=13, byrow=TRUE)

rownames(tab) <- NULL
colnames(tab) <- c('Studie', 'PG', 'EP', 'GR', 'Verteilung', 'SSE', '$\\alpha$',
                   '$\\theta$', '$\\lambda_1$', '$\\lambda_2$', '$\\lambda_3$',
                   '$\\vartheta$', '$\\psi$')

results <- as.data.frame(tab)

# Speichern
write.table(results, "results_popma.txt", sep = "\t", row.names = FALSE)

# Funktion zur Überprüfung von NA-Werten für Zeichenketten und numerische Werte
is_non_empty <- function(x) {
  return(!is.na(x) & x != "")
}

# Spalten mit mindestens einem nicht-NA-Wert ermitteln
nicht_leere_spalten <- colSums(sapply(results, is_non_empty)) > 0

# Konvertieren Sie die Tabelle in eine Markdown-Tabelle
print(results[, nicht_leere_spalten])
```

```
##      Studie   PG   EP   GR Verteilung          SSE     $\\lambda_1$
## 1 Medtronic NiRi TSmF TAVR     M. W&E 6.267981e-08 3.44497568075907
## 2 Medtronic NiRi TSmF SAVR     M. W&E 6.242942e-08  2.0164017738128
##       $\\lambda_2$      $\\vartheta$          $\\psi$
## 1 60.1381595480143 0.253223444765152 3.68803883356448
## 2 108.529717338558  2.39262794823416 3.46086592396405
```